LETTER **Communicated by Omri Barak**

Differential Geometry Methods for Constructing Manifold-Targeted Recurrent Neural Networks

Federico Claudi
federicoclaudi@protonmail.com
Tiago Branco
t.branco@ucl.ac.uk
Sainsbury Wellcome Centre for Neural Circuits and Behaviour, University College London, London W1T 4JG, U.K.

Neural computations can be framed as dynamical processes, whereby the structure of the dynamics within a neural network is a direct reflection of the computations that the network performs. A key step in generating mechanistic interpretations within this computation through dynamics framework is to establish the link among network connectivity, dynamics, and computation. This link is only partly understood. Recent work has focused on producing algorithms for engineering artificial recurrent neural networks (RNN) with dynamics targeted to a specific goal manifold. Some of these algorithms require only a set of vectors tangent to the target manifold to be computed and thus provide a general method that can be applied to a diverse set of problems. Nevertheless, computing such vectors for an arbitrary manifold in a high-dimensional state space remains highly challenging, which in practice limits the applicability of this approach. Here we demonstrate how topology and differential geometry can be leveraged to simplify this task by first computing tangent vectors on a low-dimensional topological manifold and then embedding these in state space. The simplicity of this procedure greatly facilitates the creation of manifold-targeted RNNs, as well as the process of designing task-solving, on-manifold dynamics. This new method should enable the application of network engineering–based approaches to a wide set of problems in neuroscience and machine learning. Our description of how fundamental concepts from differential geometry can be mapped onto different aspects of neural dynamics is a further demonstration of how the language of differential geometry can enrich the conceptual framework for describing neural dynamics and computation.

Federico Claudi is the corresponding author.

Neural Computation **34, 1790–1811** (2022)
https://doi.org/10.1162/neco_a_01511

1 Introduction

Networks of neurons can be viewed as dynamical systems in which the joint activity of all units is a state that represents the information stored in the network and its dynamics represent computations (Vyas, Golub, Sussillo, & Shenoy, 2020; Sussillo, 2014). Under this computation through dynamics perspective, understanding neuronal computation requires describing the dynamics of neural networks and how these are determined by their connectivity structure (Schaeffer, Khona, Meshulam, International Brain Laboratory, & Fiete, 2020; Finkelstein et al., 2021). Neural dynamics are often conceptualized as trajectories in an n-dimensional vector space—the state space—in which the distance along each dimension represents the firing rate of individual neurons. Recent work suggests that in both biological and artificial neural networks, such neural trajectories are often confined to a lower-dimensional subspace of state space (Gao & Ganguli, 2015; Gao et al., 2017; Russo et al., 2018; Gallego, Perich, Miller, & Solla, 2017; Maheswaranathan, Williams, Golub, Ganguli, & Sussillo, 2019; Khona & Fiete, 2021; Beiran, Meirhaeghe, Sohn, Jazayeri, & Ostojic, 2021) and occasionally to a neural manifold with additional topological structure (Kim, Rouault, Druckmann, & Jayaraman, 2017; Gardner et al., 2021; Chaudhuri, Gerçek, Pandey, Peyrache, & Fiete, 2019). It has been further hypothesized that the geometry and topology of these low-dimensional neural manifolds are linked in a fundamental way to the computations carried out by the network (Maheswaranathan et al., 2019; Gao et al., 2017; Jazayeri & Ostojic, 2021; Darshan & Rivkind, 2022; Chung & Abbott, 2021; Pollock & Jazayeri, 2020; Duncker & Sahani, 2021). This view therefore emphasizes that understanding how network connectivity gives rise to structured neural dynamics is a key goal toward explaining neural computations.

Achieving this goal will likely require the measurement of activity and connectivity of large numbers of neurons spanning multiple brain regions in individual animals. While recent technological developments have enabled simultaneous activity recordings from hundreds of neurons and detailed reconstructions of network connectivity (Steinmetz et al., 2021; Stringer et al., 2019; Winnubst et al., 2019; Osten & Margrie, 2013), a complete description of a network's structure and activity in behaving animals remains largely beyond the reach of experimental neuroscience. On the other hand, artificial recurrent neural networks (RNNs) can be trained to solve a variety of tasks similar to those employed in experimental neuroscience, and their connectivity structure and dynamics are perfectly known. This makes them an ideal testing ground for developing theoretical and analytical tools to investigate the links among connectivity, dynamics, and computation (Sussillo, 2014; Sussillo & Barak, 2013; Mastrogiuseppe & Ostojic, 2018). The majority of work employing RNNs to address these issues uses tools from dynamical systems theory to reverse-engineer the neural dynamics of networks trained to perform a task (Sussillo & Barak, 2013;

Schaeffer et al., 2020). This optimization-based approach allows RNNs to discover how to best structure their dynamics to carry out a certain computation. An alternative approach is to develop general algorithms for directly constructing networks with dynamics that are targeted to a preselected manifold in an effort to produce a deeper understanding of how connectivity determines dynamics Recent work in this direction has led to methods for engineering low-rank RNNs with targeted dynamics (Mastrogiuseppe & Ostojic, 2018; Beiran, Dubreuil, Valente, Mastrogiuseppe, & Ostojic, 2020), RNNs displaying manifold attractor dynamics (Darshan & Rivkind, 2022), and general algorithms for producing manifold-targeted RNNs (Pollock & Jazayeri, 2020; Biswas & Fitzgerald, 2020). Similarly, the neural engineering framework (NEF) outlined in Eliasmith and Anderson (2004) aims to develop a general, theory-grounded framework for engineering targeted neural networks (Barak & Romani, 2021; Eliasmith, 2005).

Briefly, the NEF provides a framework for mapping low-dimensional dynamics to high-dimensional neural representations of external variables through the definition of encoding, decoding, and dynamics functions. The analytical methods used to derive the decoding function, however, depend on understanding the relationship between stimuli and neural activity (Humphries, 2003). When this relationship is complex, applying the NEF is a challenging task. In addition, NEF is constrained to a specific manifold set by a linear mapping followed by point-wise nonlinearity, and alternative methods are required for engineering RNNs targeted to a wider class of manifolds.

Another recently developed method for RNN engineering that is particularly promising is the embedding manifolds with population-level Jacobians (EMPJ; Pollock & Jazayeri, 2020), which enables the creation of RNNs with dynamics confined to a target manifold in state space. Briefly, EMPJ takes a set of vectors tangent to the target manifold and builds a system of equations that yields the connectivity matrix for a network with dynamics that lay on the target manifold. Tangent vectors play a dual role in EMPJ: by being tangent to the target manifold, they confine the constructed RNN dynamics to it, and their orientation and magnitude dictate the direction and speed of the on-manifold (i.e., alongside the manifold's surface) RNN dynamics. Thus, producing specific on-manifold dynamics with EMPJ requires that a set of precisely oriented tangent vectors is identified (e.g., tangent vectors at points in the neighborhood of an attractor state should be oriented toward that state to drive the dynamics toward it). However, computing the required tangent vectors is complicated by the fact that for a sufficiently high-dimensional state space, there is an infinite number of ways to embed a low-dimensional manifold, and the position and orientation of the tangent vectors depend on the choice of embedding (see Figure 1A). Furthermore, the tangent vectors need to be computed anew when a different embedding is selected or if the dimensionality of the state space is changed (e.g., to produce networks with different numbers of units), requiring

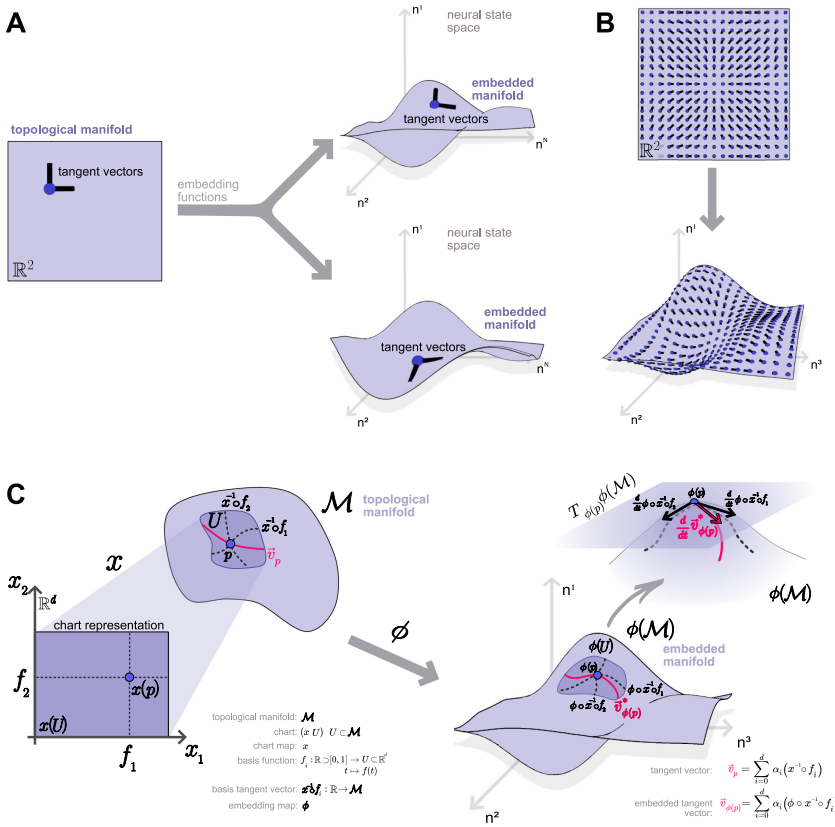

Figure 1: Topological manifolds and tangent vectors. (A) Left: The topological manifold $\mathbb{R}^2$ with a schematic representation of tangent vectors at a point. Right: Embeddings of $\mathbb{R}^2$ in $\mathbb{R}^3$. (B) Tangent vector fields. A tangent vector field on the topological manifold $\mathbb{R}^2$ (top) and embedded in $\mathbb{R}^3$ (bottom). (C) Tangent vectors. Left: A topological manifold $\mathcal{M}$ and a chart (x, U) containing a point p with tangent vectors through it. The basis functions f_i through $x(p)$ in the chart's local coordinates system $x(U)$ are shown. Right: Visualization of the manifold and tangent vectors embedded in $\mathbb{R}^3$ (bottom) and the relationship between tangent vectors as curves through a point on the manifold and n-dimensional vectors in the tangent plane (top).

additional effort for computing tangent vectors. Altogether, these properties restrict the set of problems to which EMPJ can easily be applied to those for which computing tangent vectors on the target manifold is relatively simple.

In this work, we aim to address this limitation of EMPJ by demonstrating how tools from topology and differential geometry can be used to facilitate the task of computing tangent vectors. Specifically, we describe how to compute the tangent vectors on a topological manifold prior to its embedding in a high-dimensional state space. This new approach offers two key advantages. First, on-manifold dynamics (e.g., position and number of fixed points), which are specified by the relative orientation and magnitude of the tangent vectors, are easier to conceptualize on the canonical topological manifold rather than in its embedding in a higher-dimensional state space (see Figure 1B). Indeed, they are generally chosen to match the (often low-dimensional) dynamics of latent task variables (Pollock & Jazayeri, 2020; Kao, 2019; Jazayeri & Ostojic, 2021; Maheswaranathan et al., 2019) and are thus independent of the embedding state-space dimensionality. Second, once tangent vectors have been computed on a topological manifold, the corresponding vectors on any embedding of the manifold can be computed with minimal additional effort. This simplifies the task of computing tangent vectors for EMPJ and reducing the requirements for additional computation when the manifold embedding or embedding space is changed. While the techniques used here are well established in their respective fields and routinely used in others (e.g., physics and engineering), they are relatively new to neuroscience. We believe that our work shows how ideas from differential geometry map naturally onto neural dynamics concepts and are therefore a powerful framework for understanding the link among connectivity, dynamics, and computation. We outline these ideas in detail as a means of introducing them to the neuroscience community (see the online mathematical appendix) and showcase their utility by computing tangent vectors used to create manifold targeted RNNs.

2 Computing Tangent Vectors

In this section, we demonstrate the application of concepts from differential geometry to the task of computing tangent vectors for creating manifold targeted RNNs. We leave the precise definitions of key objects, highlighted in italics, to the mathematical appendix and instead focus on providing an intuitive understanding of the method.

2.1 Tangent Vectors on the Topological Manifold. A *topological manifold* $\mathcal{M}$ is a *topological space* locally homeomorphic to Euclidean space $\mathbb{R}^d$ (e.g., the plane and the sphere are locally similar to $\mathbb{R}^2$ at every point). In neuroscience, neural manifolds are often conceptualized as being situated or embedded in a higher-dimensional vector space $\mathbb{R}^n$, the neural state space. A topological manifold, however, does not necessarily exist within a larger space. Indeed, computations are often more easily carried out on the manifold prior to its embedding, as in the case of tangent vectors. For

clarity, here we refer to a manifold embedded in state space as an embedded manifold and as a topological manifold otherwise, although technically both are topological manifolds. A topological manifold is simply defined by a set and a *topology*. For example, the line $\mathbb{R}^1$ manifold can be represented by the set $[0, 1]$ endowed with the *standard topology*. We leave the definition of each of the manifolds used in this work to the section 5.

For a d-dimensional manifold $\mathcal{M}$ embedded in an n-dimensional space (with $n > d$), *tangent vectors* are n-dimensional vectors tangent to $\mathcal{M}$ at a point (see Figure 1C). This view of tangent vectors, however, depends on the manifold being embedded in a larger vector space and thus cannot be used as a general definition of a tangent vector on a topological manifold. Instead, tangent vectors at a point $p \in \mathcal{M}$ are defined as equivalence classes of parameterized curves $\gamma : \mathbb{R} \supset I \to \mathcal{M}$ such that two curves are equivalent if they share the same directional derivative at p. This more abstract definition of tangent vectors is equivalent to the traditional view of tangent vectors for embedded manifolds (see below), but it enables us to compute tangent vectors on the topological manifold directly.

Any tangent vector v_p belongs to a tangent vector space at $p : T_pM$ and can thus be defined as a linear combination of a set of basis vectors of T_pM: a set of linearly independent vectors $e_i \in T_pM$ spanning T_pM. A fundamental theorem in differential geometry establishes that $dim(T_pM) = dim(\mathcal{M})$ such that d-many basis vectors need to be defined to obtain a complete basis set of T_pM. The construction of the basis of T_pM relies on the concept of a *chart* of a topological manifold: a chart establishes a local coordinates system by mapping an open neighborhood $U \subset \mathcal{M}$ to a subset of $\mathbb{R}^d$ using a bijective map x (see Figure 1C). For a point p, it is then possible to determine its position $x(p)$ in the local coordinates system defined by a chart. In the same coordinates system, it is possible to define d-many parameterized functions f_i going through $x(p)$ and parallel to one axis of the local coordinates system. These can then be projected onto the manifold as $x^{-1} \circ f_i$, thus producing a set of parameterized functions on the manifold with different directional derivatives that can act as representative functions for basis vectors of T_pM (see Figure 1C). Let $e_i = [x^{-1} \circ f_i]$ be a basis vector; then any tangent vector v_p can be expressed as a linear combination of the basis vectors, $v_p = \sum_{i=1}^{d} \alpha_i e_i$, where the α_i are scalar factors.

Computing a tangent vector of a topological manifold thus requires (1) definition of the manifold itself, (2) construction of set of charts covering the manifold, (3) definition of the basis functions f_i, and (4) definition of the scalar factors α_i. We discuss the final step in more detail below, but for now, we note that steps 1 to 3 need only be carried out once per topological manifold. Once this has been done for a manifold, recomputing tangent vectors for different factors α_i requires only carrying out simple calculations (which can be implemented in computer code) but no additional analytical work. Indeed, tangent vectors on any embedding of the topological manifold can also be effortlessly computed, as we show next.

2.2 Tangent Vectors on the Embedded Manifold. The *embedding* of a topological manifold can intuitively be conceptualized as placing the manifold within another, higher-dimensional, manifold or, as in this case, in a vector space, such that the resulting embedded manifold is still a topological manifold (i.e., without tears or self-intersections). For a given sufficiently high-dimensional embedding space, a manifold can be embedded in infinitely many ways, resulting in embedded manifolds with very different geometries (see Figure 1A). Importantly, all embedded manifolds share the same topological structure as a result of being an embedding of the same topological manifold (e.g., any embedding of the plane is always an open two-dimensional surface in a higher-dimensional space). Nevertheless, the widely different geometry of the embedded manifolds results in differently oriented tangent vectors, complicating the task of identifying tangent vectors for RNN design (see Figure 1A).

This difficulty can be avoided simply by noting that given tangent vectors computed on a topological manifold as described above and an embedding function ϕ, ϕ can be used to easily compute the corresponding tangent vectors on the embedded manifold. It is in fact possible to define a *pushforward* map $\phi^* : T_pM \to T_{\phi(p)}\phi(M)$ assigning to each element v_p of the tangent vector space at a point on $\mathcal{M}$ an element $v^*_{\phi(p)}$ of the corresponding tangent vector space on the embedded manifold. An alternative equivalent approach is to project the basis vectors of T_pM, $e_i = [x^{-1} \circ f_i]$ onto the embedded manifold as $e^*_i = [\phi \circ x^{-1} \circ f_i]$ (see Figure 1C). Then, $v^*_{\phi(p)} = \sum_{i=1}^{d} \alpha_i e^*_i$, where the α_i are the same scalar factors as above. The $v^*_{\phi(p)}$ thus defined are valid tangent vectors according to the definition of tangent vectors as equivalence classes above. Creating manifold targeted RNNs, however, requires tangent vectors in the familiar form of n-dimensional vectors at a point to build a system of equations that can be solved to obtain the network's connectivity (Pollock & Jazayeri, 2020). These can be obtained by taking the derivative of the map $\phi \circ x^{-1} \circ f_i$ and evaluating it at $\phi(p)$, which can be done numerically since $\phi \circ x^{-1} \circ f_i$ is a parameterized curve in $\mathbb{R}^n$ (it is a map $\mathbb{R} \to \mathbb{R}^n$). The n-dimensional basis vectors e^*_i obtained through the derivative operation can then be combined to give the n-dimensional vector form of v^*_p used for RNN fitting (see Figure 3A).

Most manifolds of interest in neuroscience are low dimensional, and for these, embedding maps onto $\mathbb{R}^3$ are easy to define. For example, for the two-dimensional manifold S^2 (the sphere), $\phi(p) = (\sin(p_0) * (p_1), \sin(p_0) * \sin(p_1), \cos(p_0))$ is an embedding (given an appropriate definition of the manifold's set; see section 5) as it gives the familiar unit sphere centered at the origin. In general, however, embedding maps for arbitrarily large embedding spaces, like the ones of interest in neuroscience, are harder to define. To overcome this limitation, we note that an orthonormal $(k \times n)$ matrix N acts as an embedding of $\mathbb{R}^k$ in $\mathbb{R}^n$, for $k < n$. Thus, if an embedding ϕ of $\mathcal{M}$ into $\mathbb{R}^k$ exists, it is possible to embed $\mathcal{M}$ in $\mathbb{R}^n$ as $N\phi(\mathcal{M})$. This

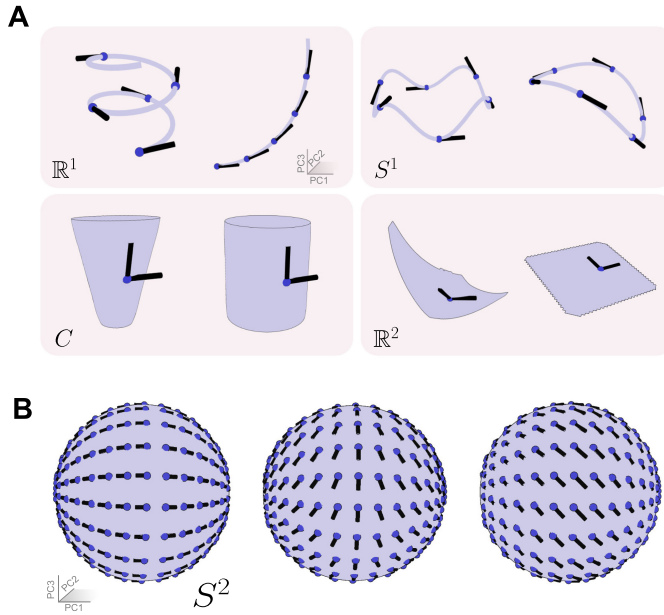

Figure 2: Tangent vectors on embedded manifolds. (A) Tangent vectors (black) at selected points (blue dots) on one- and two-dimensional manifolds (light blue lines and surfaces) embedded in $\mathbb{R}^{64}$ visualized in PCA space. The tangent vectors on the embedded manifold were computed with the procedure outlined in this work. (B) Tangent vector fields on embeddings of the sphere in $\mathbb{R}^{64}$ visualized in PCA space. $\mathbb{R}^{64}$ was selected as an arbitrary high-dimensional embedding space. Tangent vectors produced by three different tangent vector fields on the sphere manifold are shown (see section 5 for details).

procedure can therefore be used to embed manifolds onto arbitrarily large state spaces, with the limitation that only embeddings that are possible in a lower-dimensional space can be used, therefore reducing the range of geometries of the embedded manifold. Figure 2A shows embeddings of one- and two-dimensional manifolds in an arbitrarily high-dimensional vector space $\mathbb{R}^n$ (with $n = 64$; visualized in three-dimensional PCA space) obtained with this procedure. Importantly, the method for computing tangent vectors described here is agnostic to the way that embedding maps are obtained and can thus be used with embedding maps obtained through other methods.

2.3 Tangent Vector Fields. In the preceding sections, we demonstrated how a tangent vector can be defined as a linear combination of basis vectors of T_pM or $T_{\phi(p)}\phi(M)$, which requires that d-many scalar coefficients α_i

be specified. The choice of coefficients determines the orientation and magnitude of the tangent vector and thus the on-manifold dynamics of the RNN at that point, as discussed in section 1. To obtain the desired dynamics, it is necessary to specify different coefficients for each point on the manifold to produce vectors with different orientations and magnitudes. This can be achieved by defining a map $\psi : \mathcal{M} \to \mathbb{R}^d$ assigning a d-dimensional vector to each point on the manifold whose elements are the α_i coefficients. Thus, ψ effectively assigns a tangent vector to each $p \in \mathcal{M}$, making it a *tangent vector field* and different choices of ψ define different vector fields on the same manifold (see Figure 2B). Importantly, ψ is defined on the topological manifold such that a given vector field map can be used to generate RNNs with the same on-manifold dynamics but different embedded manifold geometry.

Thus, a vector field producing specific on-manifold dynamics (e.g., by specifying the number and location, on the manifold, of fixed points) will produce dynamics that will look different in state space (due to the different embeddings of the manifold) but with the same on-manifold dynamics (see Figures 1A and 1B).

On-manifold dynamics can therefore be defined on a d-dimensional space independent of the number of units in the network (which determines the dimensionality of the state space) and how the n-dimensional dynamics unfold in state space (which depends in part on the choice of embedding function). Conceptually, ψ captures the latent variables dynamics solving a given task, which often depends on the logical structure of the task itself (e.g., on the number of latent variables and their dynamics; Pollock & Jazayeri, 2020; Jazayeri & Ostojic, 2021; and Maheswaranathan et al., 2019, though see Guest & Martin, 2021, for a discussion about multiple realizability).

3 Constructing Targeted RNN

The main aim of this work is to define a procedure to simplify the computation of tangent vectors on target manifolds. In the preceding section, we described how differential geometry allows for simple computation of tangent vectors on embedded manifolds for any choice of embedding function ϕ or tangent vector field function ψ given a topological manifold $\mathcal{M}$. In this section, we demonstrate that the tangent vectors computed as described above can be used to produce RNNs with the desired dynamics. Pollock and Jazayeri (2020) described a method for obtaining an RNN's connectivity matrix from a set of tangent vectors. Here we propose a similar alternative procedure for achieving the same goal.

The RNNs used here are simple, autonomous, dynamical systems of n identical units whose dynamics are defined as

$$\dot{h} = W\sigma(h), \tag{3.1}$$

where $\dot{h}$ represents the first derivative of the network's state h with respect to time, W is an $n \times n$ matrix whose entries define the strength of the connection between two units, and σ is a nonlinear function (here *tanh*). Since the RNN dynamics are entirely specified by W, once an initial condition is selected, the task of creating manifold targeted RNNs can be reduced to finding a W yielding the desired dynamics when the network's state is initialized on the target manifold.

The network state corresponds to a point in the n-dimensional state space. For an RNN whose dynamics are confined to a target (embedded) manifold, this implies $h = \phi(p) \in \phi(\mathcal{M})$. Similarly, $\dot{h}$ represents a velocity vector indicating how fast and in which direction the network state will evolve and must therefore be tangent to the target manifold at all times. Thus, $\dot{h}$ is a tangent vector of the embedded manifold ($\dot{h} = v_p^* \in T_{\phi(p)}\phi(\mathcal{M})$; see Figure 3A). Using these identities, we can then rewrite equation 3.1 as

$$v_p^* = W\sigma(\phi(p)), \tag{3.2}$$

to reflect the notation described in the previous section.

Given a point on the topological manifold and embedding map ϕ and vector field map ψ, we can compute $\phi(p)$ and v_p^* (see Figure 3A) such that equation 3.2 can be solved for W. However, equation 3.2 has n knowns and n^2 unknowns. Thus, in practice, we sample k-many points on the topological manifold and build a system of equations,

$$\begin{bmatrix} v_{p_1}^* \\ v_{p_2}^* \\ \vdots \\ v_{p_k}^* \end{bmatrix} = W \begin{bmatrix} \sigma(\phi(p_1)) \\ \sigma(\phi(p_2)) \\ \vdots \\ \sigma(\phi(p_k)) \end{bmatrix}, \tag{3.3}$$

which can be used to solve for W using least squares. The choice of k (the number of sampling points) does not generally affect the results as long as the manifold is sampled with sufficient density. Manifolds with more curvature and more rapidly varying vector fields require a larger number of sampling points. The appropriate number for each target dynamics can be determined empirically.

To evaluate the performance of our method, we defined tangent vector fields on embeddings of one- and two-dimensional manifolds in a state space of arbitrary large dimensionality ($\mathbb{R}^n$, with $n = 64$; see Figures 3B and 3C). Next, we used the procedure outlined in the previous section to obtain RNNs targeted to the embedded manifolds and with dynamics specified by the tangent vectors (see section 5). To determine how well the resulting RNN dynamics were confined to the target manifold, we evaluated the off-manifold drift of RNNs targeted to the sphere and cylinder

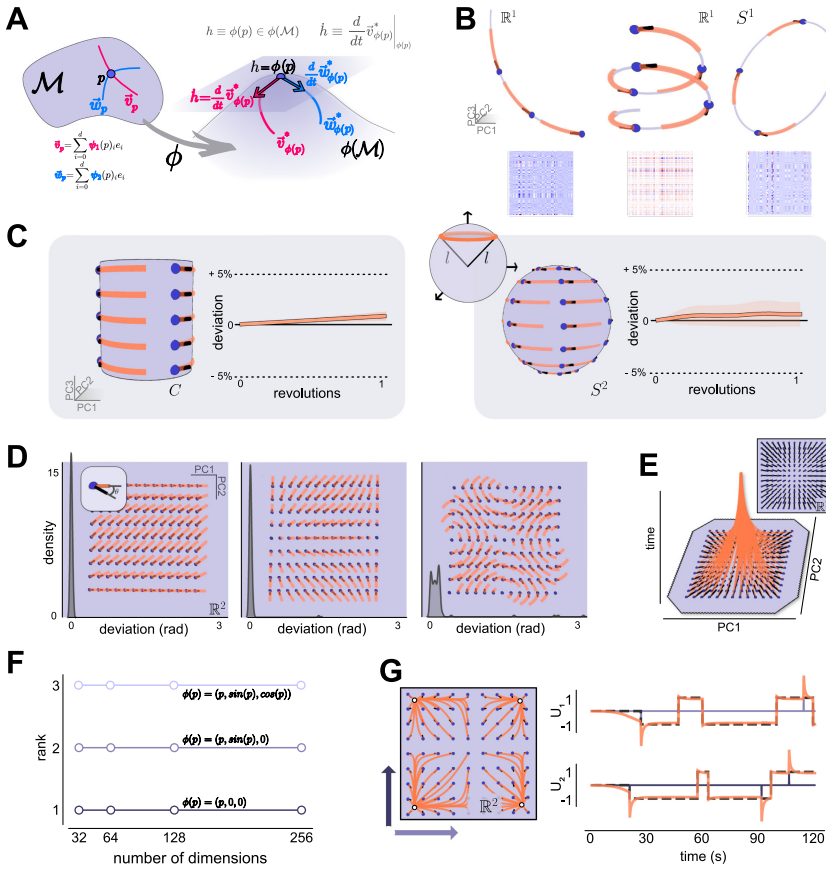

Figure 3: Quantification of RNN dynamics. (A) Schematic illustrating the relationship between ψ, ϕ, p, h, $\dot{h}$, and $\vec{v}_p$. (B) RNN dynamics (salmon traces) of RNNs fitted to one-dimensional manifolds embedded in $\mathbb{R}^{64}$ visualized in PCA space. Insets show the connectivity matrices of fitted RNNs. (C) Quantification of dynamics drift. Inset: Schematic illustration of experiment setup. The dynamics were chosen such that on-manifold dynamics would evolve along trajectories at a constant distance from the origin. Panels shows the dynamics (salmon) of RNNs fitted to the cylinder (left) and sphere (right) embedded in $\mathbb{R}^{64}$ visualized in PCA space and the quantification of deviation from the manifold surface (as a percentage of the initial distance) of the RNN dynamics over one revolution around the manifold (mean and standard deviation averaged across repeats with RNNs initialized at different points on the manifold). (D) Quantification of dynamics accuracy. Salmon RNN dynamics—black target vector field on the plane manifold embedded in $\mathbb{R}^{64}$ visualized in 2D PCA space. The black kernel density estimates show the distribution of angular difference between RNN dynamics and tangent vector field (see inset; see section 5 for more details).

manifolds. Briefly, the target manifold and tangent vectors were chosen such that on-manifold dynamics would unfold along trajectories at a constant distance from the origin of state space (see Figure 3C, inset). We were thus able to quantify off-manifold dynamics by measuring how the RNNs' neural state's distance from the origin changed over time, and we found minimal off-manifold dynamics (1.00 ± 0.008 % and 0.996 ± 0.096 % of the normalized distance for the plane and sphere manifolds respectively; see Figure 3C). Empirically, the magnitude of the off-manifold dynamics drift depends on several interacting factors, including the quality of the fit and the choices for activation function, embedding map, and vector field. This analysis shows that for an appropriate choice of embedding map and vector field map, our method yields tangent vectors that can be used to produce manifold targeted RNNs. Furthermore, the same tangent vectors can be used with methods such as EMPJ (Pollock & Jazayeri, 2020) to produce more accurately targeted dynamics.

To determine whether the on-manifold dynamics of RNNs constructed through our procedure was precisely determined by the orientation of tangent vectors used for the construction of the RNN, we constructed RNNs targeted to the plane manifold embedded in a high-dimensional state space. The RNN dynamics were specified by three different vector fields defined over the plane. To measure how accurately RNN dynamics followed the vector field, we measured the angle between RNN dynamics and tangent vectors at various points on the manifold and found these to be small, indicating that the RNNs' dynamics matched the vector fields with high-fidelity (0.011 ± 0.02, 0.022 ± 0.13, and 0.124 ± 0.31 radians for the three target vector fields, respectively; see Figure 3D). We next assessed how the rank of the RNNs' connectivity matrix reflected the choice of embedding map and the dimensionality of the target embedding space. We measured the rank for RNNs targeted to three different embeddings of the line manifold in state spaces of varying dimensionality. We found that the connectivity matrices' rank depended exclusively on the choice of embedding function and was

(E) Fixed points dynamics. Dynamics of an RNN fitted to the plane manifold embedded in $\mathbb{R}^{64}$ with a tangent vector field (black lines, inset) producing an attractor state at the center of the manifold. The visualization shows the manifold in two-dimensional PCA space with time on the third axis to illustrate the evolution of the RNN dynamics. (F) Rank of the connectivity matrix of networks of different dimensions fitted to three different embeddings of $\mathbb{R}^1$. (G) Task solving RNN dynamics. Left: Two-dimensional PCA visualization of the dynamics of an RNN fitted to $\mathbb{R}^2$ embedded in $\mathbb{R}^{64}$ with the tangent vector field shown in the figure. Arrows indicate the orientation of the RNN input vectors u_1 and u_2. Right: Example trial. Purple lines show the two task inputs, dotted lines the expected outputs. Solid orange lines show the outputs of the RNN. See section 5 for more details.

not affected by the dimensionality of the target space (see Figure 3F). Different embeddings of the one-dimensional line manifold produced embedded manifolds with different minimal embedding dimensions, and the rank of the neural networks' connectivity reflected that.

The approach described here can also be used to create networks displaying attractor dynamics through the appropriate choice of vector fields (see Figure 3D). We exploited this property to create a network capable of solving a two-bit memory task (adapted from Sussillo & Barak, 2013), by engineering an RNN targeted to the plane manifold and with four attractor states on it (see Figure 3G, left). The resulting network was capable of integrating the transient task inputs correctly by ignoring redundant inputs and using correct ones to switch between attractor states, thus correctly solving the task (mean squared error between expected output and RNN predictions: 0.035; see Figure 3G). In summary, these results show that tangent vectors computed with the procedure outlined in this work can be used to engineer manifold targeted RNNs.

4 Discussion

In this letter, we have leveraged topology and differential geometry to simplify the task of computing vectors tangent to a manifold in state space. These vectors can be used with local-geometry-based RNN engineering algorithms (Pollock & Jazayeri, 2020) to create RNNs with dynamics that unfold along the target manifold and have rich, task-solving, on-manifold dynamics (see Figure 3). Our approach facilitates the computation of tangent vectors by carrying it out on a topological manifold prior to embedding it in state space. This has the advantage that once a tangent vector is defined on the topological manifold, the corresponding vector can be computed on any embedding of the manifold in state space.

By using the language of differential geometry, we can separately define the geometry of the embedded neural manifold (which depends on the embedding function) and the on-manifold dynamics (specified by a vector field over the topological manifold) of an RNN targeted to it. Conceptually, the topology of a network's activity manifold and its on-manifold dynamics are determined by the logical structure of the task being performed and are shared across networks with different architectures (Maheswaranathan et al., 2019; Pollock & Jazayeri, 2020; Jazayeri & Ostojic, 2021; Maheswaranathan et al., 2019). Nevertheless, the geometry of the embedded manifolds is constrained by the properties of the individual network (i.e., choice of hyperparameters) and largely independent of the computation being performed. For example, networks using ReLU as the nonlinear activation function can produce accurate manifold targeted dynamics only when the manifold is embedded in the positive range of state space. On the other hand, networks using *tanh* require that the embedded manifold is symmetric with respect to the origin. The choice of embedding map

therefore influences the accuracy of the dynamics of the fitted RNNs, and specific embedding may be chosen for different purposes. One may use an embedding map aimed at producing neural dynamics that reproduce experimental recordings, or different embedding maps may be tested to assess their effects on neural dynamics and computation (e.g., resilience to noise). Our procedure for generating manifold targeted networks is flexible enough to work with any choice of network architecture and embedding function. This should facilitate the engineering of targeted RNNs and enable future work investigating the links between manifold topology, embedding, and network properties, as well as how these affect neural dynamics and computation. Importantly, differential geometry provides a language for describing manifold topology and geometry independently, thus mirroring the independent nature of these phenomena in artificial neural networks. In biological systems, a separation between the constraints imposed by the task requirements and those imposed by the network properties may not necessarily be found. Nevertheless, we believe that the adoption of differential geometry would still be useful for conceptualizing the relationships between computations and network properties, as well as for providing a tool for describing and comparing the geometry of neural manifolds.

The intrinsic topology of neural manifold and on-manifold dynamics is increasingly regarded as crucial for understanding neural computations (Jazayeri & Ostojic, 2021; Darshan & Rivkind, 2022; Chung & Abbott, 2021). Having a conceptual framework for exploring these ideas is thus crucial for gaining further insights into the link among network connectivity, dynamics, and computation. Given the intrinsically geometric nature of network dynamics, we believe that differential geometry should be a fundamental element of such framework. It allows for deep and precise understanding of several key concepts in neural dynamics (e.g., manifold topology versus embedded geometry), thereby providing a promising venue for the abstraction of fundamental mechanistic explanations of computation across neural networks with different properties. The approach presented here is a step in applying topology and differential geometry-based approaches to investigate the link of connectivity, dynamics, and computation, as well as to enable the application of methods such as EMPJ (Pollock & Jazayeri, 2020) to a wider class of problems in neuroscience and machine learning.

5 Methods

All work presented in this letter was carried out using custom Python code. The code, including scripts to replicate all figures, is available at the GitHub repository: https://github.com/FedeClaudi/manyfolds. The Python code makes use of open source science software Python packages including NumPy, scikit, vedo, and matplotlib (Harris et al., 2020; Pedregosa et al., 2011; Hunter, 2007; Musy et al., 2022).

5.1 Manifolds.

5.1.1 Manifold Definitions. Topological manifolds are defined by a choice of set M and topology. Here we assume the standard topology throughout. The following sets were used to define each manifold ($\times$ defines the Cartesian product):

- Line $\mathbb{R}^1$: $M = [0, 1]$.
- Circle S^1: $M = [0, 2\pi]$.
- Plane $\mathbb{R}^2$: $M = [0, 1] \times [0, 1]$.
- Cylinder C: $M = [0, 2\pi] \times [0, 1]$.
- Sphere S^2: $M = [0, \pi] \times [0, 2\pi]$.

5.1.2 Manifold Embeddings. All manifolds were embedded in $\mathbb{R}^{64}$ except for Figure 3F. The embedding was achieved in two steps as described in the text: a function $\phi : \mathcal{M} \to \mathbb{R}^3$ was used to embed the manifolds in $\mathbb{R}^3$, and a random orthonormal 3×64 matrix acted as an embedding map $\mathbb{R}^3 \to \mathbb{R}^{64}$.

The following embedding functions ϕ were used:

- $\mathbb{R}^1$ as helix.
  $\phi(p) = (\frac{\cos(4*\pi*p)}{2}, \frac{\sin(4*\pi*p)}{2}, p + 0.25)$
- $\mathbb{R}^1$ as line.
  $\phi(p) = (\sin(2p) - 0.5, 2\sin(p) - 1, -4\cos(p) + 3)$
- S^1 as curved circle.
  $\phi(p) = (\sin(p), 0.8\cos(p), \frac{\cos(2p)^2}{2} + 0.5)$
- S^1 as bent circle.
  $\phi(p) = (\sin(p), 0.8\cos(p), \frac{\cos(p)^2}{2} + 0.5)$
- C as cone.
  $\phi(p) = (\frac{k\sin(p_0)}{2}, \frac{k\cos(p_0)}{2}, p_1 + 0.1)$.
  $k = \frac{p_1}{2} + 0.4$
- C as cylinder.
  $\phi(p) = (\frac{\sin(p_0)}{2}, \frac{\cos(p_0)}{2}, p_1 + 0.1)$
- $\mathbb{R}^2$ as curved plane.
  $\phi(p) = 2(p_0, \sin(p_1), 0.4(p_1 - p_0)^2)$
- $\mathbb{R}^2$ as flat plane.
  $\phi(p) = (p_0 + 0.2, p_1 + 0.2, \frac{(p_0+p_1)}{2})$
- S^2 as unit sphere.
  $\phi(p) = (\sin(p_0)\cos(p_1), \sin(p_0)\sin(p_1), \cos(p_0))$

5.1.3 Visualizing Embedded Manifolds. Three-dimensional visualizations of embedded manifolds and RNN dynamics were realized with the Python package vedo (Musy et al., 2022). A set of m (25–1000) points was sampled from the topological manifold, and the coordinates in embedding space were computed. Then a PCA model was fitted using the scikit package (Pedregosa et al., 2011), and the first three principal components were used

to reduce the dimensionality of the point cloud to three dimensions. The manifold's surface was reconstructed in PCA space using algorithms implemented in vedo (Musy et al., 2022). For Figure 3E, the first two principal components were used, and the third dimension represented time.

5.2 Manifold Charts and Basis Functions. To define a set of charts covering the topological manifold, one or two charts per manifold were used. When two charts were used, the chart set U_i and map x_i were carefully selected such that $x_i(U_i)$ was homeomorphic to the same set in $\mathbb{R}^d$ for each chart, simplifying the definition of basis functions. The basis functions were simply defined as maps $f_i : [0, 1] \to x_i(U_i)$ parallel to one axis of the local coordinate space. For instance, for a two-dimensional manifold with $x_i(U_i) \cong [0, 1] \times [0, 1]$, the first basis function for a point $p \in \mathcal{M}$ is defined as $f_1(\lambda) = (\lambda, x(p))$.

The following charts and basis functions were used:

- For $\mathbb{R}^1$, a single chart (x, U) was defined with $U = M$ and $x := x(p) \to 2p$.
- For S^1, two charts were used with $U_1 = [0, \pi]$, $U_2 = [\pi, 2\pi]$ and $x_1 := x(p) = p$, $x_2 := x(p) = p - \pi$.
- For $\mathbb{R}^2$, a single chart was used with $U = M$ and $x := x(p) = p$.
- For S^1, two charts were used with $U_1 = [0, \pi] \times [0, \pi]$, $U_2 = [0, \pi] \times [\pi, 2\pi]$ and $x_1 := x_1(p) = (\frac{p_0}{\pi}, \frac{p_1}{\pi})$, $x_2 := x_2(p) = (\frac{p_0}{\pi}, \frac{p_1 - \pi}{\pi})$.
- For C, two charts were used with $U_1 = [0, \pi] \times [0, 1]$, $U_2 = [\pi, 2\pi] \times [0, 1]$ and $x_1 := x_1(p) = (\frac{p_0}{\pi}, p_1)$ and $x_2 := x_2(p) = (\frac{p_0 - \pi}{\pi}, p_1)$.

5.3 Vector Fields. Tangent vector fields maps (ψ) were used to specify the magnitude and orientation of tangent vectors. For Figure 2B, the following vector field maps were used:

- $\psi(p) = (-\frac{sign(\cos(p_0))}{3}, 0)$.
- $\psi(p) = (-\lambda\frac{\cos(p_0)}{2}, \frac{\lambda}{4})$ with $\lambda = 1.5 - abs(\cos(p_0))$.
- $\psi(p) = (\frac{1}{4}, \frac{1}{4})$.

For Figure 3B, the vector field was $\psi(p) = 1$, while in panel C, the vector field used was $\psi(p) = (0, 1)$ for the sphere and $\psi(p) = (1, 0)$ for the cylinder. In panel D, the three vector fields were:

- $\psi(p) = \frac{1}{3}(\sin(p_0 p_1), 1)$.
- $\psi(p) = \frac{1}{3}(\sin(\pi(p_0 - \frac{1}{2}))\cos(\pi(p_1 - \frac{1}{2})))$.
- $\psi(p) = \frac{1}{3}(\sin(2\pi p_1), \sin(2\pi p_0))$.

For panel E, $\psi(p) = 3(\frac{1}{2} - p_0, \frac{1}{2} - p_1)$. The vector field for panel G is described in section 5.9.

5.4 Tangent Vectors Computation. To compute the tangent vector at a point $\phi(p)$ on the embedded manifold, the position $x_i(p)$ of the point in the chart representation was computed (given a chart (x_i, U_i) such that $p \in U_i$). Next, the basis functions were defined as described above to obtain a one-dimensional curve in $x_i(U_i) \subset \mathbb{R}^d$ for each basis. Next, the projection of the basis functions onto the embedded manifold was computed as $\phi \circ x^{-1} \circ f_i$, yielding one-dimensional curves in $\mathbb{R}^n$. The derivative of these curves with respect to the parameter λ of f_i was computed and evaluated at the point $\phi(p)$ to obtain the basis tangent vector. Finally, the tangent vector field map ψ was evaluated at p to obtain the coefficients for the linear combination of basis vectors and thus compute the tangent vector.

5.5 RNN Creation. The procedure for generating manifold targeted RNNs is described in the main text. Here, we note that for each manifold, a set of $k = 3 - 100$ equally spaced points was used to build the system of equations whose solution, obtained via least squares using NumPy, gave the RNN connectivity matrix W.

5.6 Dynamics Drift. To estimate how much the RNN dynamics drifted off the surface of the target embedded manifolds over time, 10 RNNs were fitted to the cylinder and sphere manifolds. The choice of manifold, embedding function, and tangent vector fields was such that on-manifold dynamics would evolve along trajectories equidistant from the origin at all times. Thus, off-manifold deviation could simply be determined by measuring changes in the distance between the network state and the origin. Each RNN was initialized at 25 different locations on the embedded manifold and allowed to evolve until it completed an entire revolution around the manifold and returned to its original position. The average and standard deviations of the normalized distance from the origin of state space for all RNNs and all starting positions on a given manifold were computed and are shown in Figure 3C.

5.7 Dynamics Accuracy. To estimate how accurately RNN dynamics matched the direction prescribed by tangent vector fields, three different vector fields on the flat plane embedded manifold were used, as described above. For each, 10 RNNs were fitted to it and initialized at 81 different locations on the manifold, from where they were allowed to evolve for the simulation time equivalent of 5 seconds. The angle between the vector from the initial location and the final RNN state and the tangent vector at the initial location was computed for each initialization. The distribution of angular differences is shown in Figure 3D.

5.8 RNN Matrix Rank. To estimate the rank of manifold targeted RNNs' connectivity matrices (see Figure 3F), RNNs with different number

of units ($n = 32$, 64, 128, and 256) were fitted to the line manifold $\mathbb{R}^1$ in $\mathbb{R}^n$ with one of the following embedding maps:

- $\phi_1(p) = (p, 0, 0)$, a straight line,
- $\phi_1(p) = (p, \sin(p), 0)$, a planar curve,
- $\phi_1(p) = (p, \sin(p), \cos(p))$, a three-dimensional curve,

to produce embedded manifolds with different extrinsic dimensionalities. The rank of the connectivity matrix W of each RNN was then estimated in Python, using the NumPy library.

5.9 Task-Solving Dynamics.

5.9.1 2-Bit Memory Task Design. The 2-bit memory task was created by adapting the 3-bit memory task used in other work (Sussillo & Barak, 2013; Maheswaranathan et al., 2019). In brief, two independent and time-varying inputs were produced. At each time step, the inputs could assume values of 1, -1, or 0. The task's goal consists of keeping track of the last nonzero value received from each input. Given two inputs and two possible values for each, four possible combinations are possible. The network must keep an internal representation of the current combination and update it correctly in response to new nonzero inputs. This includes ignoring redundant inputs (e.g., two consecutive 1s), which should not change the network's internal representation.

5.9.2 RNN Design. As the starting point for designing the task-solving RNN dynamic, we used the solution discovered by training networks on the 3-bit memory task (Sussillo & Barak, 2013; Maheswaranathan et al., 2019). The presence of two latent variables (the "state" of each input) naturally suggests the use of a two-dimensional manifold, and the absence of a periodic structure in the inputs suggests that a plane would be ideal. For simplicity, we embedded $\mathbb{R}^2$ in $\mathbb{R}^n$ ($n = 64$) as a flat plane by (1) selecting a random unit norm n-dimensional vector (v_0), (2) selecting a second random vector orthogonal to the first (v_1), and (3) using the embedding function $\phi(p) = \frac{1}{5}(p_0 v_0 + p_1 v_1) - 0.5$. This yielded a target manifold consisting of a randomly oriented square manifold centered at the origin. The scaling factor $\frac{1}{5}$ can constrain the dynamics closer to the origin where they evolve more rapidly, and the manifold scale could be chosen to adjust the timescale of the dynamics.

The requirement that the network be able to maintain four stable states in the absence of stimuli suggests that four fixed-point attractors should be included in the on-manifold dynamics. Stimuli should push the dynamics from one attractor state to another to change the network's internal memory, but repeated stimuli should not push the network state outside the current attractor's basin of attraction. To achieve this, the following tangent vector field was used: $\psi(p) = 0.6(-\sin(2.5\pi(p_0 - 0.1)), -\sin(2.5\pi(p_1 - 0.1)))$.

The RNN dynamics equation 3.1 does not include external inputs. To create a task-solving RNN we modified equation 3.1 to

$$\dot{h} = W\sigma(h) + Bu,$$

where u is the two-dimensional inputs vector and B is the $(n \times 2)$ input matrix describing how each input affects each unit in the network. We defined

$$B := \begin{bmatrix} | & | \\ v_0 & v_1 \\ | & | \end{bmatrix}$$

such that each input moved the network along one of the "sides" of the plane manifold.

The network output was a two-dimensional vector defined as $y = B^{-1}h$, thus translating the position along each "side" of the embedded plane into a scalar output.

Acknowledgments

We thank Francesca Mastrogiuseppe for fruitful discussions about this project.

Authors' Contribution

F.C. designed the study and performed all analysis. F.C. and T.B. wrote the manuscript.

References

Barak, O., & Romani, S. (2021). Mapping low-dimensional dynamics to high-dimensional neural activity: A derivation of the ring model from the neural engineering framework. *Neural Comput.*, *33*(3), 827–852. 10.1162/neco_a_01361, PubMed: 33513322

Beiran, M., Dubreuil, A., Valente, A., Mastrogiuseppe, F., & Ostojic, S. (2020). Shaping dynamics with multiple populations in low-rank recurrent networks. *Neural Computation*, *33*, 1572–1615. 10.1162/neco_a_01381

Beiran, M., Meirhaeghe, N., Sohn, H., Jazayeri, M., & Ostojic, S. (2021). *Parametric control of flexible timing through low-dimensional neural manifolds*. bioRxiv:2021.11.08.467806.

Biswas, T., & Fitzgerald, J. E. (2020). *A geometric framework to predict structure from function in neural networks.* arXiv:2010.09660.

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., & Fiete, I. (2019). The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nat. Neurosci.*, *22*(9), 1512–1520. 10.1038/s41593-019-0460-x, PubMed: 31406365

Chung, S., & Abbott, L. F. (2021). Neural population geometry: An approach for understanding biological and artificial neural networks. *Current Opinion in Neurobiology*, *70*, 137–144. 10.1016/j.conb.2021.10.010, PubMed: 34801787

Darshan, R., & Rivkind, A. (2022). Learning to represent continuous variables in heterogeneous neural networks. *Cell Reports*, *39*, 110612. 10.1016/j.celrep.2022.110612, PubMed: 35385721

Duncker, L., & Sahani, M. (2021). Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Curr. Opin. Neurobiol.*, *70*, 163–170. 10.1016/j.conb.2021.10.014, PubMed: 34837752

Eliasmith, C. (2005). A unified approach to building and controlling spiking attractor networks. *Neural Comput.*, *17*(6), 1276–1314. 10.1162/0899766053630332

Eliasmith, C., & Anderson, C. H. (2004). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. Cambridge, MA: MIT Press.

Finkelstein, A., Fontolan, L., Economo, M. N., Li, N., Romani, S., & Svoboda, K. (2021). Attractor dynamics gate cortical information flow during decision-making. *Nat. Neurosci.*, *24*(6), 843–850. 10.1038/s41593-021-00840-6, PubMed: 33875892

Gallego, J. A., Perich, M. G., Miller, L. E., & Solla, S. A. (2017). Neural manifolds for the control of movement. *Neuron*, *94*(5), 978–984. 10.1016/j.neuron.2017.05.025, PubMed: 28595054

Gao, P., & Ganguli, S. (2015). On simplicity and complexity in the brave new world of large-scale neuroscience. *Curr. Opin. Neurobiol.*, *32*. 148–155. 10.1016/j.conb.2015.04.003, PubMed: 25932978

Gao, P., Trautmann, E., Yu, B., Santhanam, G., Ryu, S., Shenoy, K., & Ganguli, S. (2017). *A theory of multineuronal dimensionality, dynamics and measurement*. bioRxiv:10.1101/21462.

Gardner, R. J., Hermansen, E., Pachitariu, M., Burak, Y., Baas, N. A., Dunn, B. J., . . . Moser, E. I. (2021). Toroidal topology of population activity in grid cells. *Nature*, *602*, 123–128. 10.1038/s41586-021-04268-7

Guest, O., & Martin, A. E. (2021). *On logical inference over brains, behaviour, and artificial neural networks*. PsyArXiv. 10.31234/osf.io/tbmcg

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., . . . Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. 10.1038/s41586-020-2649-2, PubMed: 32939066

Humphries, M. D. (2003). Book review. *Conn. Sci.*, *15*(2–3), 141–143. 10.1080/09540090310001598448

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science Engineering*, *9*(3), 90–95. 10.1109/MCSE.2007.55

Jazayeri, M., & Ostojic, S. (2021). *Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity*. arXiv:2107.04084.

Kao, J. C. (2019). Considerations in using recurrent neural networks to probe neural dynamics. *J. Neurophysiol.*, *122*(6), 2504–2521. 10.1152/jn.00467.2018, PubMed: 31619125

Khona, M., & Fiete, I. R. (2021). *Attractor and integrator networks in the brain.* arXiv:2112.03978.

Kim, S. S., Rouault, H., Druckmann, S., & Jayaraman, V. (2017). Ring attractor dynamics in the drosophila central brain. *Science*, *356*(6340), 849–853. 10.1126/science.aal4835, PubMed: 28473639

Maheswaranathan, N., Williams, A. H., Golub, M. D., Ganguli, S., & Sussillo, D. (2019). Universality and individuality in neural dynamics across large populations of recurrent networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems, 22* (pp. 15629–15641). Red Hook, NY: Curran.

Mastrogiuseppe, F., & Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, *99*(3), 609–623.e29. 10.1016/j.neuron.2018.07.003, PubMed: 30057201

Musy, M., Jacquenot, G., Dalmasso, G., neoglez, de Bruin, R., Pollack, A., . . . ilorevilo. (2022). marcomusy/vedo: 2022.0.1. Zenodo. 10.5281/zenodo.5842090

Osten, P., & Margrie, T. W. (2013). Mapping brain circuitry with a light microscope. *Nat. Methods*, *10*(6), 515–523. 10.1038/nmeth.2477, PubMed: 23722211

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, *12*(85), 2825–2830.

Pollock, E., & Jazayeri, M. (2020). Engineering recurrent neural networks from task-relevant manifolds and dynamics. *PLOS Comput. Biol.*, *16*(8), e1008128. 10.1371/journal.pcbi.1008128

Russo, A. A., Bittner, S. R., Perkins, S. M., Seely, J. S., London, B. M., Lara, A. H., . . . Churchland, M. M. (2018). Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, *97*(4), 953–966. 10.1016/j.neuron.2018.01.004, PubMed: 29398358

Schaeffer, R., Khona, M., Meshulam, L., International Brain Laboratory, & Fiete, I. R. (2020). Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, *33*. Red Hook, NY: Curran.

Steinmetz, N. A., Aydin, C., Lebedeva, A., Okun, M., Pachitariu, M., Bauza, M., . . . Harris, T. D. (2021). Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, *372*(6539). 10.1126/science.abf4588, PubMed: 33859006

Stringer, C., Pachitariu, M., Steinmetz, N., Reddy, C. B., Carandini, M., & Harris, K. D. (2019). Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, *364*(6437), 255. 10.1126/science.aav7893, PubMed: 31000656

Sussillo, D. (2014). Neural circuits as computational dynamical systems. *Curr. Opin. Neurobiol.*, *25*, 156–163. 10.1016/j.conb.2014.01.008, PubMed: 24509098

Sussillo, D., & Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput.*, *25*(3), 626–649. 10.1162/NECO_a_00409

Vyas, S., Golub, M. D., Sussillo, D., & Shenoy, K. V. (2020). Computation through neural population dynamics. *Annu. Rev. Neurosci.*, *43*, 249–275. 10.1146/annurev-neuro-092619-094115, PubMed: 32640928

Winnubst, J., Bas, E., Ferreira, T. A., Wu, Z., Economo, M. N., Edson, P., . . . Chandrashekar, J. (2019). Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain. *Cell*, *179*(1), 268–281.

Received October 13, 2021; accepted March 3, 2022.