Differential geometry methods for constructing manifold-targeted recurrent neural networks.

Federico Claudi[1]*, Tiago Branco[1]

Abstract

Neural computations can be framed as dynamical processes, whereby the structure of the dynamics within a neural network are 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 between 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 only require 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. Furthermore, 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.

[1] *Sainsbury Wellcome Centre for Neural Circuits and Behaviour, UCL, London, UK.*
***Corresponding author**: federico.claudi.17@ucl.ac.uk

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 et al., 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 et al., 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 and Ganguli, 2015; Gao et al., 2017; Russo et al., 2018; Gallego et al., 2017; Maheswaranathan et al., 2019), and occasionally, to a neural manifold with additional topological structure (Kim et al., 2017; Gardner et al., 2021; Chaudhuri et al., 2019). It has been further hypothesized that the geometry and topology of these low dimensional neural manifolds is linked in a fundamental way to the computations carried out by the network (Maheswaranathan et al., 2019; Gao et al., 2017; Jazayeri and Ostojic, 2021; Darshan and Rivkind, 2021; Chung and Abbott, 2021; Pollock and Jazayeri, 2020). This view therefore emphasizes that understanding how network connectivity gives rise to structured neural dynamics is a key goal towards explaining neural computations.

Achieving this goal will likely require the measurement of activity and connectivity of large number 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 and 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 between connectivity, dynamics and computation (Sussillo, 2014; Sussillo and Barak, 2013; Mastrogiuseppe and Ostojic, 2018). The majority of work employing RNNs to address these issues uses tools from dynamical systems theory to reverse-engineer the neu-

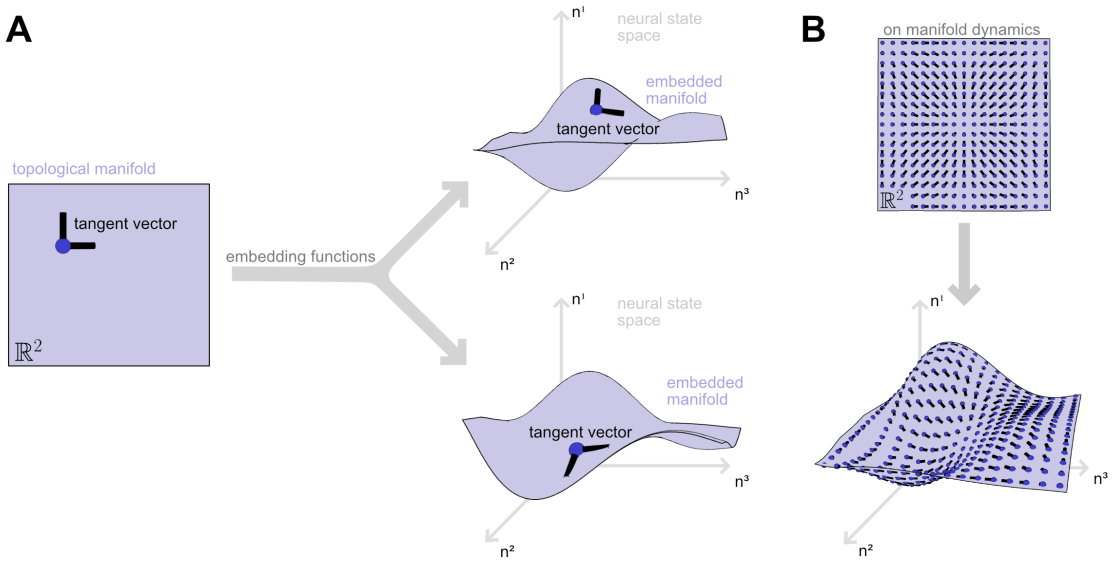

Figure 1. A. Left, the topological manifold $\mathbb{R}^2$ with a schematic representation of tangent vectors at a point. Right, two different embeddings of $\mathbb{R}^2$ in $\mathbb{R}^3$ with the corresponding tangent vectors. **B.** On manifold dynamics. Top, schematic representation of a tangent vectors field on the topological manifold $\mathbb{R}^2$. Bottom, the corresponding tangent vector field for an embedding of $\mathbb{R}^2$ in $\mathbb{R}^3$.

ral dynamics of networks trained to perform a task (Sussillo and 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 pre-selected manifold, in an effort to produce a deeper understanding of how connectivity determines dynamics (Mastrogiuseppe and Ostojic, 2018; Beiran et al., 2020; Eliasmith and Anderson, 2003; Pollock and Jazayeri, 2020; Biswas and Fitzgerald, 2020; Darshan and Rivkind, 2021). A particularly promising approach in this direction is the Embedding Manifolds with Population-level Jacobians (EMPJ) algorithm (Pollock and Jazayeri, 2020), which enables the creation of RNNs with dynamics confined to a target manifold in state space. Briefly, EMPJ uses vectors tangent to the target manifold to build 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 RNN dynamics to it, and their orientation and magnitude dictate the on-manifold dynamics. EMPJ is therefore an elegant and parsimonious algorithm capable of producing RNNs targeted to different manifolds and displaying rich on-manifold dynamics.

In practice, however, computing the tangent vectors may be far from trivial, which effectively limits the range of problems that methods like EMPJ can be applied to. For example, a target manifold of given topology (e.g., the plane) can be embedded in state space in infinite ways. This makes computing tangent vectors a challenging task since the position and orientation of tangent vectors depend on the precise geometry of the target manifold (**Figure 1**A). The problem is made even harder by the necessity to precisely orient the tangent vectors to produce the desired on-manifold dynamics (e.g., to create attractor states; Figure **Figure 1**B, bottom). Here we aim to address this challenge by describing a general approach that simplifies the task of computing tangent vectors on target manifolds and specifying the required on-manifold dynamics (Figure **Figure 1**B). We employ concepts from topology and differential geometry to show how tangent vectors can be computed on topological manifolds prior to their embedding in state space (Figure **Figure 1**A, left), thus removing the need for recomputing the tangent vectors for different choices of manifold embeddings. This approach can in principle be combined with local geometry-based algorithms such as EMPJ, thus widening the class of problems that they can be readily applied to. We believe that this work adds to the view that ideas from differential geometry map naturally onto neural dynamics concepts and are therefore a powerful framework for understanding the link between connectivity, dynamics and computation.

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 bold, to a Mathematical Appendix section, and instead focus on providing an intuitive understanding of the method.

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, like in the case of tangent vectors. For clarity, here we will 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 $M = [0, 1]$ endowed with the **standard topology**. We leave the definition of each of the manifolds used in this work to the Methods section.

For an m-dimensional manifold $\mathcal{M}$ embedded in an n-dimensional space (with $n > m$), **tangent vectors** are n-dimensional vectors tangent to $\mathcal{M}$ at a point. 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 parametrized 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 the basis vectors of T_pM. A fundamental theorem in differential topology establishes that $dim(T_pM) = dim(\mathcal{M})$ such that d-many basis vectors need to be defined. The definition of the basis of T_pM relies on the concept of **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 (**Figure 2**A). For a point p it's 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 parametrized 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 $x^{-1} \circ f_i$ giving a set of parametrized functions on the manifold with different directional derivatives which can act as representative functions for basis vectors of T_pM (**Figure 2**A). 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, 4) definition of the scalar factors α_i. We will discuss the final step in more detail below, but for now we note that steps 1-3 need only be carried out once per topological manifold. Once this has been done for a manifold, re-computing tangent vectors for different factors α_1 simply requires 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 will show next.

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, like 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 large embedding space, a manifold can be embedded in infinitely many ways, resulting in embedded manifolds with very different geometries (**Figure 1**A). 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 a 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 (**Figure 1**A).

This difficulty can be avoided simply by noting that given tangent vectors defined on a topological manifold 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_p^* of the corresponding tangent vector space on the embedded manifold. An alternative approach, however, is to project the basis vectors of TpM, $e_i = [x^{-1} \circ f_i]$ onto the embedded manifold as $e_i^* = [\phi \circ x^{-1} \circ f_i]$ (**Figure 2**B). Then, $v_p^* = \sum_{i=1}^{d} \alpha_i e_i^*$ where the α_i are the same scalar factors as above.

The e_i^* 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

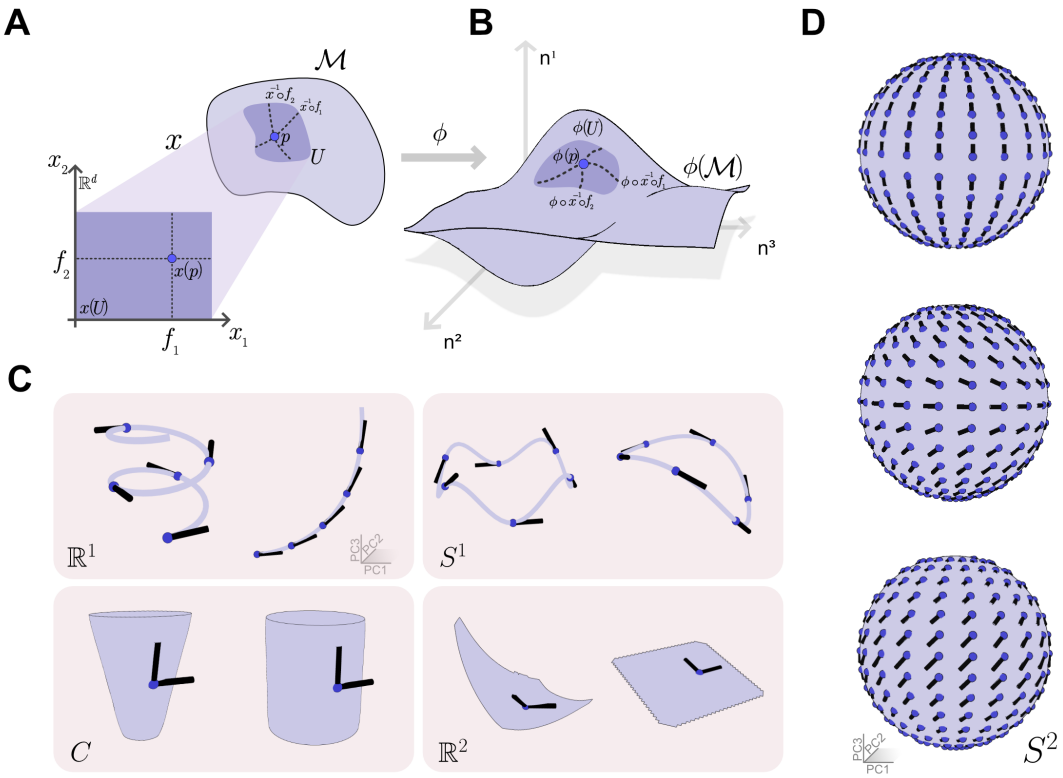

Figure 2. A. Graphic representation of a two dimensional topological manifold $\mathcal{M}$ and a chart (x, U) containing a point p. The basis functions f_i in the chart's local coordinates system are also shown. **B.** Graphic representation of an embedding of $\mathcal{M}$ with the basis functions $\phi \circ x^{-1} \circ f_i$ shown. **C.** Tangent vectors (black) at selected points (blue dots) on one and two dimensional manifolds (light blue lines and surfaces). **D.** Tangent vector fields on the sphere. The tangent vectors produced by three different tangent vector fields for the sphere manifold are shown. For panels C and D, the manifolds were embedded in $\mathbb{R}^{64}$ and visualized in three dimensions using the first three principal components of a PCA model, see Methods for details.

network's connectivity matrix. These can be obtained by taking the derivative of the map $\phi \circ x^{-1} \circ f_i$ and evaluate it at $\phi(p)$ which can be done numerically since $\phi \circ x^{-1} \circ f_i$ is a parametrized 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 be combined to give the n-dimensional vector form of v_p^* used for RNN fitting as described above.

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), $\phi(p) = sin(p_0) * cos(p_1), sin(p_0) * sin(p_1), cos(p_0))$ is an embedding (given an appropriate definition of the manifold's set, see Methods) as it gives the familiar unit sphere centered at the origin. In general, however, embedding functions 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 $(n \times k)$ 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 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. Nevertheless, this procedure can produce rich and interesting embeddings of target manifolds in state space (see **Figure 2**C for embeddings in $\mathbb{R}^{64}$). Importantly, the method for computing tangent vectors described here is agnostic to the way that embedding functions are obtained and can thus be used with embedding functions obtained through other methods.

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 are 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. To obtain the desired dynamics then it is necessary to specify

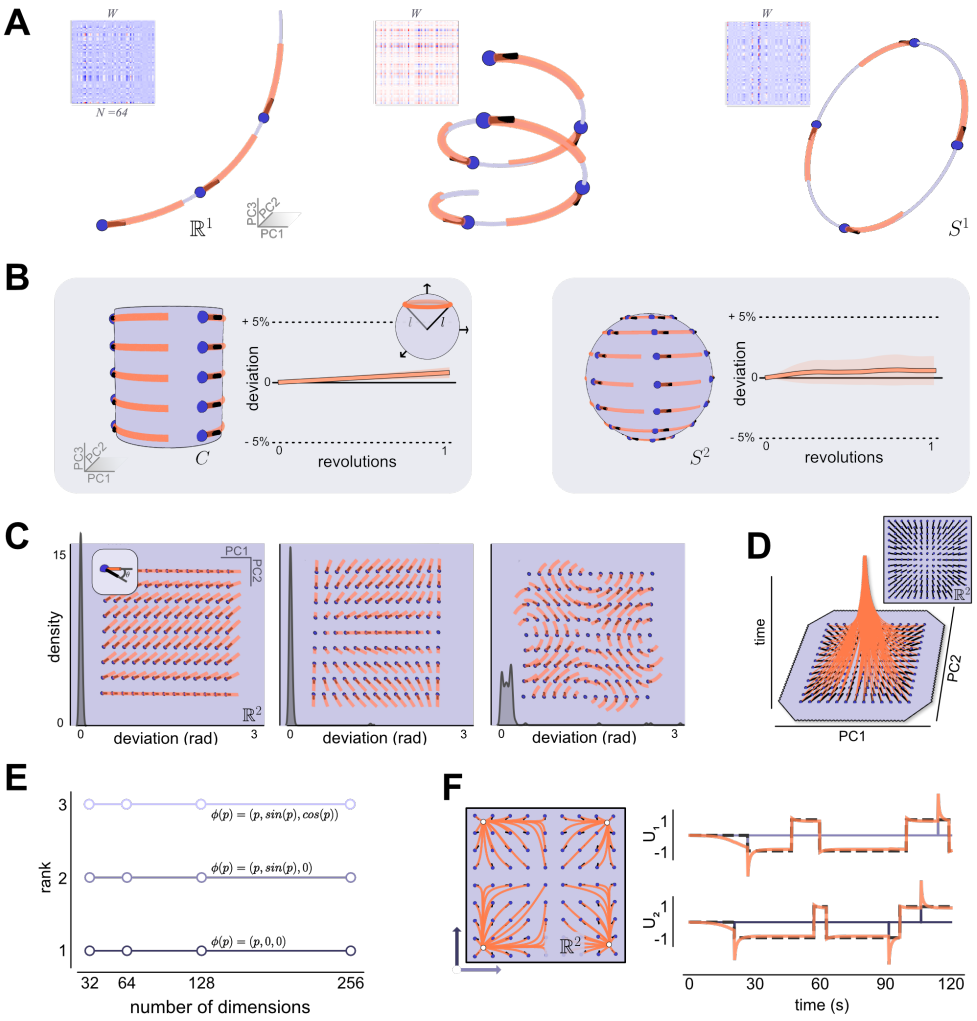

Figure 3. A. RNN dynamics (salmon traces) for RNNs targeted to one dimensional manifolds embedded in $\mathbb{R}^{64}$ visualized in PCA space (see Methods). Different traces show the RNN dynamics when the RNN was initialized to a different initial condition (blue dots). Insets show the connectivity matrix of RNNs producing the dynamics shown in figure. **B.** Quantification of dynamics drift for the cylinder (left) and sphere (right) manifolds in $\mathbb{R}^{64}$. RNNs were created with dynamics as shown on the left of each panel such that the distance from the origin of state space would remain constant over time (see inset). The plots on the right on each panel show the change in distance from the origin over one complete revolution across different starting points. **C.** Quantification of dynamics accuracy for different vector fields on the plane manifold in $\mathbb{R}^{64}$. Three different vector fields were defined for the same manifold, producing different RNN dynamics (shown in the background of each panel). The angle between the RNN dynamics and the tangent vector at the point in which the RNN was initialize was quantified (see inset) and shown as kernel density estimate distributions. **D.** Dynamics fixed points. Visualization of the dynamics of an RNN fitted using a vector field which specified a single attractor point (see inset). The plot shows the dynamics evolving in the plane spanned by the first two principal components (see Methods) over time. **E.** Connectivity matrix rank. The rank of the connectivity matrix W of networks of different dimensions fitted to three different embeddings of the line manifold in $\mathbb{R}^n$. **F.** Task solving dynamics. Left, vector field and RNN dynamics on $\mathbb{R}^2$. Arrows indicate the two inputs vectors u_1 and u_2. Right, example trial. Purple lines show the two inputs, dotted lines the expected network outputs. Solid orange lines show the actual network outputs.

different coefficients for each point on the manifold. 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 (**Figure 2**D). Importantly, ψ is defined on the topological manifold such that the same vector field map can be used to generate RNNs with the same on-manifold dynamics but different embedded manifold geometry. On-manifold dynamics can therefore be defined on a d-dimensional space independently of the number of units in the network 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 depends uniquely on the logical structure of the task itself (e.g., on number of latent variables and their dynamics; **Pollock and Jazayeri (2020); Jazayeri and Ostojic (2021); Maheswaranathan et al. (2019)**).

Constructing targeted RNN

In this section we demonstrate that tangent vectors computed with the above procedure produce manifold targeted RNNs with the desired on-manifold dynamics. While **Pollock and Jazayeri (2020)** described a method for obtaining an RNNs connectivity matrix from a set of tangent vectors, here we describe a similar alternative procedure for achieving the same goal. Next, we use these procedure to create RNNs fitted to different manifolds and with different choices of embedding and vector field maps.

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

$$\dot{h} = W\sigma(h) \tag{1}$$

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 non linear 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 $h = \phi(p)$. 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 ($\dot{h} = v_p^*$). We can then re-write **equation 1** as:

$$v_p^* = W\sigma(\phi(p)) \tag{2}$$

to reflect the notation described in the previous section.

Given a point on the topological manifold (and a vector field maps ψ) we can compute $\phi(p)$ and v_p^* such that **equation 2** can be used to compute W. However, **equation 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:

$$\begin{vmatrix} v_{p1}^* \\ v_{p2}^* \\ \vdots \\ v_{pk}^* \end{vmatrix} = W \begin{vmatrix} \sigma(\phi(p_1)) \\ \sigma(\phi(p_2)) \\ \vdots \\ \sigma(\phi(p_k)) \end{vmatrix} \tag{3}$$

which can be used to solve for W using least squares.

This simple method can be used to construct RNNs whose dynamics lay on target manifolds in arbitrarily large state spaces(**Figure 3**). The manifold targeted RNNs thus obtained have low-rank connectivity matrices (with the rank matching the embedded manifold's extrinsic dimensionality; **Figure 3**E), yet they show minimal off-manifold deviations over time (**Figure 3**B) and can display rich and accurate on-manifold dynamics (**Figure 3**C,D). Indeed the approach outlined here can be used to easily construct RNNs with task-solving dynamics **Figure 3**F. The appropriate choice of on-manifold dynamics allows the network to solve a 2-bit memory task by using four fixed points attractors to store its internal representation of the inputs. New inputs can update this representation by moving the network state between different attractors. Furthermore, the dynamics are stable with respect to repeated, redundant, inputs which do not push the state beyond the current attractor's basin of attraction. While gradient-descent algorithms are used to train networks to perform this kind of task **(Sussillo and Barak, 2013)**, here careful choice of the dynamics manifold and on-manifold dynamics yields a task-solving network directly.

Discussion

In this manuscript 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 and Jazayeri, 2020)** to create RNNs with dynamics that unfold along the target manifold and have rich, task-solving, on-manifold dynamics. 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 manifold (which depends on the embedding function) and the on-manifold dynamics (defined by a vector field over the topological manifold), as well as the manifold topology itself. 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)**. On the other hand, 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 with different non-linear activation functions (e.g.: $tahn$ vs $ReLu$) might require that the embedded manifold is confined to different sub-regions of state space. Differential geometry provides a language for describing manifold topology and geometry independently, thus mirroring the independent nature of these phenomena.

The intrinsic topology of neural manifold and on-manifold dynamics are increasingly regarded as crucial for understanding neural computations **(Jazayeri and Ostojic, 2021; Darshan and Rivkind, 2021; Chung and Abbott, 2021)**. Having a conceptual framework for exploring these ideas is thus crucial for gaining further insights into the link between 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 vs 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 between connectivity, dynamics and computation, as well as to enable the application of methods such as EMPJ **(Pollock and Jazayeri, 2020)** to a wider class of problems in neuroscience and machine learning.

Methods

Code availability

All work presented in this manuscript was carried out using custom python code. The code, included 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 and matplotlib (Harris et al., 2020; Pedregosa et al., 2011; Hunter, 2007).

Manifolds

manifolds 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:

- **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: $M = [0, \pi] \times [0, 2\pi]$

where $\times$ denotes the Cartesian product.

manifold embeddings

All manifolds were embedded in $\mathbb{R}^{64}$except for **Figure 3**E. 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**.
  $\phi(p) = (sin(2p) - 0.5, 2sin(p) - 1, -4cos(p) + 3)$
- S^1 as curved **circle**.
  $\phi(p) = (sin(p), 0.8cos(p), \frac{cos(2p)^2}{2} + 0.5)$
- S^1 as bent **circle**.
  $\phi(p) = (sin(p), 0.8cos(p), \frac{cos(p)^2}{2} + 0.5)$
- C as **cone**.
  $\phi(p) = (\frac{ksin(p_0)}{2}, \frac{kcos(p_0)}{2}, p_1 + 0.1)$.
  $k = \frac{p_1}{2} + 0.4$
- 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**.
  $\phi(p) = (sin(p_0)cos(p_1), sin(p_0)sin(p_1), cos(p_0))$

visualizing embedded manifolds

Three dimensional visualizations of embedded manifolds and RNN dynamics were realized with the python package vedo (Musy et al., 2021). A set of m (25-1000) points were sampled from the topological manifold and the coordinates in embedding space were computed. Then a PCA model model was fitted using the scikit package (Pedregosa et al., 2011) and the first three principal components was 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. For **Figure 3**D, the first two principal components were used and the third dimension represented time.

Manifold charts ad 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 $x_1 = id$, $x_2 := x(p) = p - \pi$.
- for $\mathbb{R}^2$ a single chart was used with $U = M$ and $x = id$.
- 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 $x_2 := x_2(p) = (\frac{p_0-\pi}{\pi}, p_1)$.

Vector fields

Tangent vector fields maps (ψ) were used to specify the magnitude and orientation of tangent vectors. For **Figure 2**D, 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 3**A, the vector field was $\psi(p) = 1$ while in panel **B** the vector field used was $\psi(p) = (0, 1)$ for the sphere and $\psi(p) = (1, 0)$ for the cylinder. In panel **C** 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 **D**: $\psi(p) = 3(\frac{1}{2} - p_0, \frac{1}{2} - p_1)$. The vector field for panel **F** is described in the section *Task solving dynamics* of the methods.

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.

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,

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 each using the embedding and vector field maps as described above. 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 deviation of the normalized distance from the origin of state space for all RNNs and all starting positions on a given manifold was computed and shown in figure.

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 five 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.

RNN matrix rank

To estimate the rank of manifold targeted RNNs' connectivity matrices (**Figure 3**W), 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.

Task solving dynamics

2-bit memory task design

The 2-bit memory task was created by adapting the 3-bit memory task used in other works (Sussillo and 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 in keeping track the last non-zero 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 updated it correctly in response to new non-zero inputs. This include ignoring redundant inputs (e.g., two consecutive 1s) which should not change the network's internal representation.

RNN design

As the starting point for designign the task-solving RNN dynamic, we used the solution discovered by training networks on the 3-bit memory task (Sussillo and 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, 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 n-dimensional vector (v_0), 2) selecting a second random vector orthogonal to the first (v_1) and 3) using the embedding function $\phi(p) = \frac{1}{5}(p_0 v_0 + p_1 v_1) - 0.4$.

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 1** does not include external inputs. To create a task-solving RNN we modified **equation 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 vecotr defined as $y = B^{-1}h$ thus translating the position along each 'side' if the embedded plane into a scalar output.

Mathematical Appendix

Topological manifold and topology

A **topological manifold** $\mathcal{M}$ is an Hausdorff connected topological space locally homeomorphic to $\mathbb{R}^d$. Thus, for every point $p \in \mathcal{M}$, there exists a neighborhood U with $p \in U$ such that U is homeomorphic to $\mathbb{R}^d$with d constant for every point in the manifold. Then d also specifies the dimensionality of the manifold.

A **topological space** is defined by a set of points M and a **topology** $\mathcal{O}$. The set is generally an interval $I \subset \mathbb{R}$ or the Cartesian product of d-many such intervals. Different sets can be used to define the same manifold, for instance the line $\mathbb{R}^1$ can have $M = [0, 1]$ or $M' = [1, 2]$ as underlying sets. The topology $\mathcal{O}$ is defined as a subset of the powerset of $M : (P)(M)$ such that: 1) $\emptyset, M \in \mathcal{O}$, 2) unions of elements of $\mathcal{O}$ are also in $\mathcal{O}$ and 3) intersections of elements of $\mathcal{O}$ are also in $\mathcal{O}$.

The **standard topology** is intuitively conceptualized as the topology of Euclidean space $\mathbb{R}^d$. More precisely, for every $x \in M$, and $rin\mathbb{R}^+$, define the open ball of radius r centered at x as the set $B_r(x) := y \in M | \sqrt{\sum_i^d (y^i - x^i)} < r$. Then, an open subset U is in the standard topology of $\mathbb{R}^d$if there's a ball of some, possibly small, radius contained in U.

Tangent vector

Given a parametrized curve $\gamma : \mathbb{R} \supset I \to \mathcal{M}$ and a point $p \in \mathcal{M}$, then two equivalent definitions of **tangent vectors** can be given. A tangent vector can be defined as the directional derivative operator at p along γ or, alternatively, as the equivalence class of parametrized curves through p sharing the same directional derivative. The two definitions are equivalent, there's a one-to-one relationship between the directional derivative operators and the equivalence classes, in this work we used the latter definition.

Tangent vector space

The tangent vectors at a point $p \in \mathcal{M}$ define a (tangent) vector space T_pM with well-defined operations of vector addition and scalar multiplication (technically T_pM is an algebra since an operation of vector multiplication is also defined). A fundamental result in differential geometry proves that $dim(M) = dim(T_pM)$, that is T_pM and $\mathbb{R}^d$ are isomorphic as vector spaces. Given a chart (x, U) with $p \in U$, the basis of T_pM can be constructed by defining a set of d-many curves f_i in the chart representation of U, and projecting them onto the manifold using $({}^{-1}x)$. These curves then act as representative for the equivalence classes that are the basis vectors. The basis functions are maps $f_i : \mathbb{R} \supset I \to x(U)$ defined to be parallel to the i^{th} axis of the local coordinates frame determined by the char.

Chart

A chart of a d-dimensional manifold $\mathcal{M}$ with set M is a pair (x, U) with $U \in \mathcal{P}(M)$ and $x : U \to x(U) \subset \mathbb{R}^d$ a bijective homemorphism of U onto a subset of $\mathbb{R}^d$. The existence of x is ensured by the definition of a topological manifold as locally homeomorphic to $\mathbb{R}^d$. The component functions $x^i : U \to \mathbb{R}$ are called the coordinates of p with respect to the chart (x, U) and the chart establishes a local coordinates system around p. One or more, potentially overlapping, charts can be defined such that the union of the charts' set covers M.

Embedding

An **embedding** is a map between topological manifolds $\phi : \mathcal{M} \to \mathcal{N}$ that is an **immersion** and such that the mapped manifold $\phi(\mathcal{M})$ is a submanifold of the target manifold (i.e., it's a manifold $\mathcal{L}$ whose set L is a subset of the set N of $\mathcal{N}$). An immersion is a smooth map ϕ between manifolds such that its derivative (the **pushforward** map ϕ^*) is injective at every point $p \in \mathcal{M}$. In this work, the target manifold is the vector space $\mathbb{R}^n$which is a topological manifold with additional operations of vector addition and scalar multiplication defined.

Pushforward

Given an map between manifolds $\phi : \mathcal{M} \to \mathcal{N}$, then the **pushforward** (or derivative) of ϕ, ϕ_* is a map between tangent vector spaces of the source manifolds to tangent vector spaces on the target manifold: $\phi_* : T_pM \to T_\phi N$. Given the definition of tangent vectors as equivalence classes of curves, the pushfoward of the vector $[\gamma]$, $\phi_*\gamma]$ is $[\phi \circ \gamma]_{\phi(p)}$.

Tangent vector field

Given a manifold $\mathcal{M}$, the manifold together with all the vector spaces T_pM at each point in $\mathcal{M}$ form a bundle of topological manifolds: $TM := \bigcup_{p \in M} T_pM$. A vector field, or tangent vector field, is a section ψ of the topological bundle. That is ϕ is a map $\psi : \mathcal{M} \to T_pM$ assigning to each point p on the manifold a tangent vector (an element of the tangent vector space T_pM).

References

Beiran, M., Dubreuil, A., Valente, A., Mastrogiuseppe, F., and Ostojic, S. (2020). Shaping dynamics with multiple populations in low-rank recurrent networks.

Biswas, T. and Fitzgerald, J. E. (2020). A geometric framework to predict structure from function in neural networks.

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., and 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.

Chung, S. and Abbott, L. F. (2021). Neural population geometry: An approach for understanding biological and artificial neural networks.

Darshan, R. and Rivkind, A. (2021). Learning to represent continuous variables in heterogeneous neural networks.

Eliasmith, C. and Anderson, C. H. (2003). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT Press.

Finkelstein, A., Fontolan, L., Economo, M. N., Li, N., Romani, S., and Svoboda, K. (2021). Attractor dynamics gate cortical information flow during decision-making. *Nat. Neurosci.*, pages 1–8.

Gallego, J. A., Perich, M. G., Miller, L. E., and Solla, S. A. (2017). Neural manifolds for the control of movement. *Neuron*, 94(5):978–984.

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

Gao, P., Trautmann, E., Yu, B., Santhanam, G., Ryu, S., and others (2017). A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*.

Gardner, R. J., Hermansen, E., Pachitariu, M., Burak, Y., Baas, N. A., Dunn, B. J., Moser, M.-B., and Moser, E. I. (2021). Toroidal topology of population activity in grid cells.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

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

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

Kim, S. S., Rouault, H., Druckmann, S., and Jayaraman, V. (2017). Ring attractor dynamics in the drosophila central brain. *Science*, 356(6340):849–853.

Maheswaranathan, N., Williams, A. H., Golub, M. D., Ganguli, S., and Sussillo, D. (2019). Universality and individuality in neural dynamics across large populations of recurrent networks. *Adv. Neural Inf. Process. Syst.*, 2019:15629–15641.

Mastrogiuseppe, F. and Ostojic, S. (2018). Linking connectivity, dynamics, and computations in Low-Rank recurrent neural networks. *Neuron*, 99(3):609–623.e29.

Musy, M., Jacquenot, G., Dalmasso, G., neoglez, de Bruin, R., Pollack, A., Claudi, F., Badger, C., icemtel, Sullivan, B., Hrisca, D., Volpatto, D., Schlömer, N., Zhou, Z.-Q., and ilorevilo (2021). marcomusy/vedo: 2021.0.2.

Osten, P. and Margrie, T. W. (2013). Mapping brain circuitry with a light microscope. *Nat. Methods*, 10(6):515–523.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(85):2825–2830.

Pollock, E. and Jazayeri, M. (2020). Engineering recurrent neural networks from task-relevant manifolds and dynamics. *PLoS Comput. Biol.*, 16(8):e1008128.

Russo, A. A., Bittner, S. R., Perkins, S. M., Seely, J. S., London, B. M., Lara, A. H., Miri, A., Marshall, N. J., Kohn, A., Jessell, T. M., Abbott, L. F., Cunningham, J. P., and Churchland, M. M. (2018). Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, 97(4):953–966.e8.

Schaeffer, R., Khona, M., Meshulam, L., International Brain Laboratory, and Fiete, I. R. (2020). Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice.

Steinmetz, N. A., Aydin, C., Lebedeva, A., Okun, M., Pachitariu, M., Bauza, M., Beau, M., Bhagat, J., Böhm, C., Broux, M., Chen, S., Colonell, J., Gardner, R. J., Karsh, B., Kloosterman, F., Kostadinov, D., Mora-Lopez, C., O'Callaghan, J., Park, J., Putzeys, J., Sauerbrei, B., van Daal, R. J. J., Vollan, A. Z., Wang, S., Welkenhuysen, M., Ye, Z., Dudman, J. T., Dutta, B., Hantman, A. W., Harris, K. D., Lee, A. K., Moser, E. I., O'Keefe, J., Renart, A., Svoboda, K., Häusser, M., Haesler, S., Carandini, M., and Harris, T. D. (2021). Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539).

Stringer, C., Pachitariu, M., Steinmetz, N., Reddy, C. B., Carandini, M., and Harris, K. D. (2019). Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437):255.

Sussillo, D. (2014). Neural circuits as computational dynamical systems. *Curr. Opin. Neurobiol.*, 25:156–163.

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

Vyas, S., Golub, M. D., Sussillo, D., and Shenoy, K. V. (2020). Computation through neural population dynamics. *Annu. Rev. Neurosci.*, 43:249–275.

Winnubst, J., Bas, E., Ferreira, T. A., Wu, Z., Economo, M. N., Edson, P., Arthur, B. J., Bruns, C., Rokicki, K., Schauder, D., Olbris, D. J., Murphy, S. D., Ackerman, D. G., Arshadi, C., Baldwin, P., Blake, R., Elsayed, A., Hasan, M., Ramirez, D., Dos Santos, B., Weldon, M., Zafar, A., Dudman, J. T., Gerfen, C. R., Hantman, A. W., Korff, W., Sternson, S. M., Spruston, N., Svoboda, K., and 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.e13.