

# Interpretable statistical representations of neural population dynamics and geometry

Adam Gosztolai,<sup>1\*</sup> Robert L. Peach,<sup>2,3</sup> Alexis Arnaudon,<sup>4</sup>  
Mauricio Barahona,<sup>4</sup> Pierre Vanderheyne<sup>1</sup>

<sup>1</sup>Signal Processing Laboratory (LTS2), EPFL, Lausanne, 1016, Switzerland

<sup>2</sup>Department of Neurology, University Hospital Würzburg, Würzburg, 97070, Germany

<sup>3</sup>Department of Brain Sciences, Imperial College London, London, W12 0NN, United Kingdom,

<sup>4</sup>Department of Mathematics, Imperial College London, London, SW7 2AZ, United Kingdom

\*To whom correspondence should be addressed; E-mail: adam.gosztolai@epfl.ch.

## Abstract

The dynamics of neuron populations during diverse behaviours evolve on low-dimensional manifolds. However, it remains challenging to disentangle the role of manifold geometry and dynamics in encoding task variables. Here, we introduce an unsupervised geometric deep learning framework for representing non-linear dynamical systems based on statistical distributions of local dynamical features. Our method provides geometry-aware or geometry-agnostic representations for robustly comparing dynamical systems based on sparse measurements. Our representations are generalisable to compare computations across systems, interpretable to discover a geometric correspondence between neural dynamics and kinematics in a primate reaching task, and intrinsically encode temporal information to give rise to a decoding algorithm with state-of-the-art accuracy. Our results suggest that using the manifold structure over temporal information is important to develop better decoding algorithms and assimilate data across experiments.

## Introduction

Despite the widespread recognition that behaviour is underpinned by the dynamics of neural populations<sup>1,2</sup>, interpreting the structure of these dynamics in the context of relevant computations remains a fundamental challenge<sup>3</sup>. Several works have focused on the geometry of neural manifolds—low-dimensional smooth subspaces of neural state space, over which the high-dimensional neural activities evolve<sup>4–12</sup>—while others have suggested that computations are encoded by the dynamical flows of neural population activity<sup>3,11,13</sup>. In the latter viewpoint, manifold geometry is merely a dynamical imprint that changes over time or across animals<sup>4</sup>. Recently, high-fidelity techniques provided simultaneous experimental access to the dynamics of large neuron populations<sup>14–16</sup> and behaviour<sup>17–19</sup>. However, theoretical frameworks for finding representations of neural dynamics that are comparable across experiments, interpretable based on behavioural variables and can independently consider the geometry and dynamics are currently lacking.

Current approaches for representing neural dynamics do not meet the above challenges. While dynamical models can accurately predict trajectories for a single experiment<sup>20–25</sup>, their implicit dependence on the manifold geometry hinders their ability to generalise to other experiments. Although alignment by fitting additional linear transformations can partially address this shortcoming<sup>24,25</sup>, this only holds for manifolds with negligible curvature. Manifold learning approaches, on the other hand, can effectively unfold non-linear structures but cannot decouple dynamics and geometry as they treat trajectories as point clouds<sup>26,27</sup>. Finally, topological data analysis methods<sup>28</sup> measure invariant structures in the data, such as loops<sup>6</sup> and tori<sup>9</sup>, while disregarding continuous differences in both

the dynamics and geometry. To overcome these limitations, we need methods that can represent non-linear dynamics *intrinsically* on the manifold.

We propose a theoretical framework with an associated computational method, called MARBLE, that combines ideas from empirical dynamical modelling<sup>29</sup> and the statistical descriptions of collective systems<sup>30,31</sup> to represent non-linear dynamics intrinsically on the manifold. Unlike in dynamical modelling<sup>20–25</sup> and current geometric deep learning methods<sup>32–37</sup>, which learn a single mathematical object to encode the dynamics globally, we focus on the distribution of local vector fields. Specifically, leveraging continuity over the manifold we use geometric deep learning<sup>38–40</sup> and unsupervised contrastive learning<sup>41</sup> to produce a similarity-preserving feature embedding of local vector fields into a shared latent space. This allows us to define a robust similarity metric between possibly sparsely and irregularly sampled dynamical systems to infer continuous and abrupt dynamical changes with respect to varying system parameters or input gains. Further, making the latent features rotation invariant obtains geometry-agnostic embeddings that facilitate comparing dynamics across system realisations. This is particularly important when a direct neuron-to-neuron comparison is unviable as in the case of RNN instances or animals. The similarity-preserving property of the embeddings is fundamental for their interpretability in terms of population-level system properties, such as decision variables and movement kinematics. For example, in RNNs trained on a contextual decision-making task, we predicted the discrimination threshold from a dynamical transition detected during gain modulation. Additionally, we discovered a geometric representation of electrophysiological recordings in macaques during a reaching task, which revealed a previously unreported spatiotemporal structure mirroring the unseen kinematics. We leveraged this structure to decode kinematic trajectories with state-of-the-art accuracy. Our results suggest that differential geometric notions can reveal as yet unaccounted-for non-linear variations in neural data that can further our understanding of neural dynamics underpinning behaviour and guide the design of brain-machine interfaces.

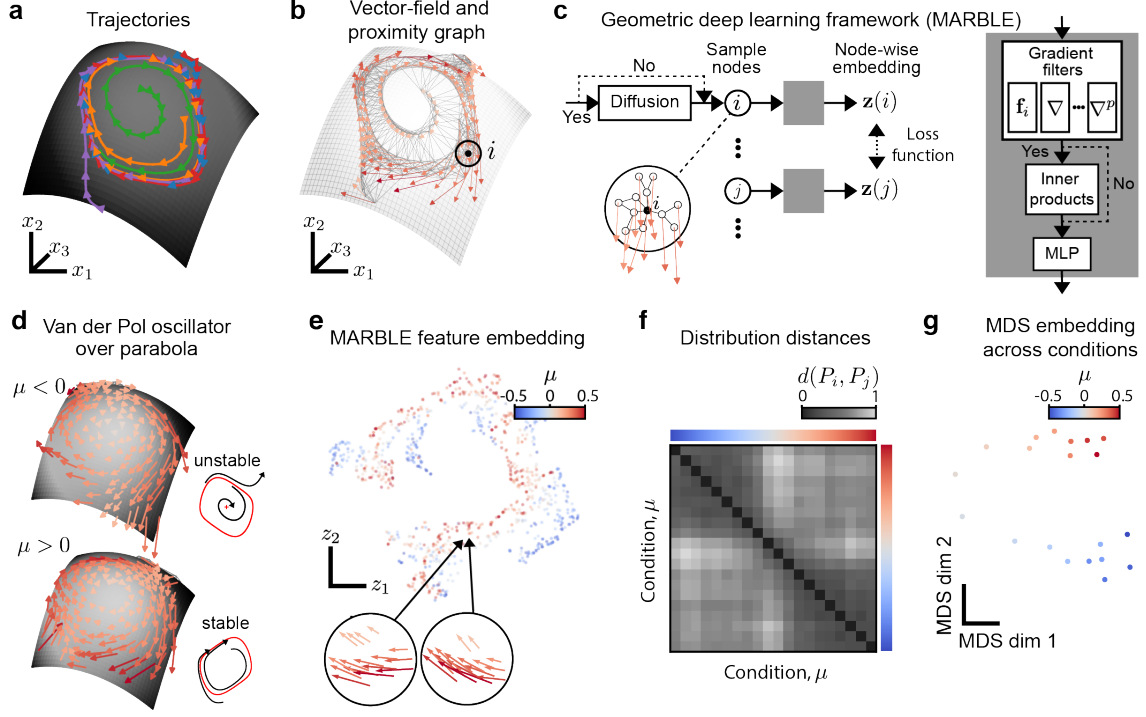
## Results

### Data-driven distributional representation of vector fields over manifolds

To characterise neural computations during behaviour, e.g., decision-making or motor actions, a typical experiment involves a set of trials producing  $d$ -dimensional time-series measurements  $\{\mathbf{x}(t)\}$  of the state of an animal, e.g., neural activity or behavioural kinematics. Frequently, these experiments are replicated under diverse conditions, requiring comparing sets of trajectories to reveal alterations in neural mechanisms. However, two main challenges hinder this analysis. First, trajectories initialised differently across trials produce uneven and possibly sparse sampling of the relevant basins of attraction of the dynamics. Second, similar dynamics may unfold over manifolds with different geometries or orientations across subjects<sup>4,24</sup>, making a direct neuron-to-neuron comparison infeasible.

To address these challenges, instead of learning the temporal evolution of the states, we build a statistical representation of the dynamics based on the local spatial context of sample points. This is possible as the trajectories of neural dynamics<sup>13</sup> or more generally of dissipative dynamical systems<sup>42</sup> converge to an  $m$ -dimensional submanifold of the original  $d$ -dimensional state-space (Fig. 1a). We first treat the trajectories under a given experimental condition  $c$  as a vector field  $\mathbf{F}_c = (\mathbf{f}_1, \dots, \mathbf{f}_n)$  anchored to a point cloud  $\mathbf{X}_c = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  (Fig. 1b). To approximate the geometry of the underlying manifold, we fit a proximity graph to  $\mathbf{X}_c$ . The local context of a point  $i$  is then given by the vector field in the  $p$ -hop neighbourhood of  $i$ . Second, we jointly embed individual local vector fields from different conditions  $c$  into a shared latent space to obtain features  $\mathbf{Z}_c = (\mathbf{z}_1, \dots, \mathbf{z}_n)$  that represent the dynamical system as an empirical distribution  $P_c = \sum_i^n \delta(\mathbf{z}_i)$ , where  $\delta$  is the Dirac delta function. To define a metric between dynamical systems under conditions  $c$  and  $c'$ , we use the optimal transport distance (Eq. S18),  $d(P_c, P_{c'})$ , between their empirical distributions in the shared latent space, which generally outperforms entropic measures (e.g., KL divergence) when detecting complex interactions based on overlapping distributions<sup>31</sup>.

The above approach hinges upon detecting overlaps between the feature distributions of dynamical systems embedded over distinct manifolds. We develop a similarity-preserving embedding of local vector fields, called MARBLE, using an unsupervised geometric deep learning architecture consisting of four components (Fig. 1c, see Sect. S1 in Materials and Methods); an optional vector diffusion kernel with learnable parameter  $\tau$  (Eq. S3) which tunes the size of the local vector fields;  $p$  gradient filter layers whose output together with  $\mathbf{f}_i$  gives the best  $p$ -th order approximation of the local vector field



**Figure 1: Data-driven statistical representation of measured dynamics over manifolds.** **a** Trajectories measured in different trials (colours) evolving over a parabolic manifold. **b** Vector field representation of trajectories. The manifold is approximated by a  $k$ -nearest neighbour graph. Black circle marks sample point and its corresponding local vector field. **c** Geometric deep learning model, MARBLE, for the node-wise embedding of local vector fields. The vector field is smoothed by an optional vector diffusion layer. The local vector fields are approximated to  $p$ -th order by gradient filters and optionally transformed into rotation-invariant inner product features. Finally, a multilayer perceptron (MLP) performs a node-wise similarity-preserving embedding of features into a latent space. **d** Vector fields of the Van der Pol oscillator over a parabolic manifold in the unstable ( $\mu = -0.25$ ) and stable ( $\mu = 0.25$ ) regimes sampled from randomly initialised trajectories ( $n \approx 200$ ). Insets show limit cycle in red and representative trajectories from a vertical projected view. **e** Similarity-preserving latent space embedding of local vector fields (two shown in insets) for different  $\mu$  visualised in 2D via UMAP embedding. **f** Distribution distances across conditions. Clustering indicates an abrupt dynamical change at  $\mu = 0$ . **g** Two-dimensional MDS embedding of the distribution distance matrix recovers the ordering of parameter  $\mu$  over two weakly connected one-dimensional manifolds.

around  $i$  (Figs. S1–S3, Eq. S13); inner product features involving learnable linear transformations which make the gradient features rotation invariant (Figs. S4–S5, Eq. S15); and a multilayer perceptron (MLP) with learnable weights  $\omega$  that produces the latent vectors  $\mathbf{z}_i$ . Unsupervised training is possible due to the spatial continuity of local vector fields over the manifold, which favours embedding adjacent vector fields close by and those randomly sampled further apart via a soft constraint in the loss function (Eq. S17). The soft constraint allows MARBLE to identify similar features across different parts of the same manifold and across different manifolds.

The use of inner product features is optional and gives rise to two operation modes. Not using them makes MARBLE geometry-aware for obtaining maximally expressive embeddings that contain information about both the geometry and the dynamics. Meanwhile, using geometry-agnostic embeddings favours learning the variation within local vector fields regardless of their spatial orientations (Figs. S4–S5). This mode should be used whenever the manifold geometry and orientation are expected to play no role in the respective neural computation. In addition, the initial vector diffusion layer is also optional and smoothes the vector field for better handling of noisy systems whose dynamics evolve near a manifold.

## Similarity of statistical representations detects continuous and abrupt changes in global dynamics

Latent parameter variations in dynamical systems can cause both continuous changes, such as slowing down or speeding up, or abrupt, qualitative transitions, such as bifurcations. As a simple illustration of our method, we use the well-known Van der Pol oscillator that is unstable for negative damping parameter  $\mu$  and stabilises as  $\mu$  increases above zero in a Hopf bifurcation (Fig. 1d and Sect. S3.1). In addition, increasing  $|\mu|$  causes continuous deformations of the limit cycle from circular to asymmetric corresponding to slow-fast dynamics. To showcase our method on a non-trivial manifold, we additionally map this system to a parabola.

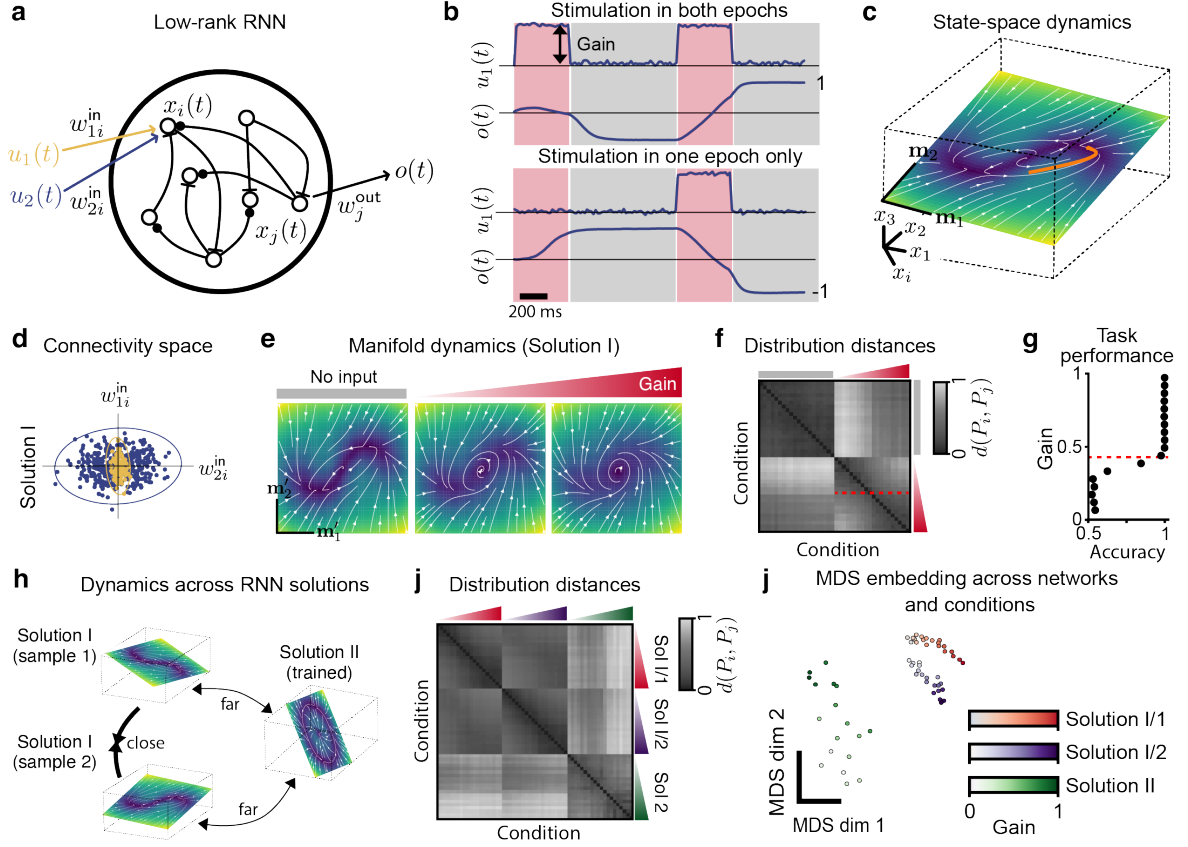
To study the dynamical response to parameter variations, we simulated short trajectories from random initial conditions for 20 different values of  $\mu$ , which plays the role of condition  $c$ , and obtained vector fields for each. Then, we trained a geometry-agnostic MARBLE network to embed the local vector fields into a common five-dimensional latent space, visualised in two dimensions via UMAP projection algorithm<sup>43</sup> (Fig. 1e). The distributional overlaps of the embedded features reflect the ability of the method to identify similar local vector fields across trajectory ensembles obtained under different conditions. For example, with larger  $|\mu|$ , the points become more disjoint, as most local neighbourhoods contain either strong convergent or divergent trajectories.

Despite the sparse and irregular sampling of the dynamics and, in turn, the local vector fields, the embedding distributions revealed robust and ordered variation in the global dynamics across  $\mu$ . Indeed, the distance  $d(P_i, P_j)$  between pairs of empirical distributions associated with conditions  $\mu_i, \mu_j$  yields a matrix with a two-partition structure for positive and negative  $\mu$  (Fig. 1f). Although this structure remained robust when we introduced manifold curvature deformations it was lost when we trained a geometry-aware network (Fig. S6) confirming the ability of MARBLE to capture on-manifold dynamics. In addition, a linear embedding of the distance matrix using multidimensional scaling (MDS) (Fig. 1g) into two dimensional space reveals a continuous deformation of the dynamics across conditions based on the ordered variation of  $\mu$  over a one dimensional manifold. This order is unexpected since it emerges from the distributional changes of irregularly spaced local vector fields. MARBLE can therefore distinguish abrupt (stable/unstable) and continuous (change of  $|\mu|$ ) deformation of dynamical systems on manifolds from a collection of sparse, irregular trajectories.

## Discriminating computational mechanisms in recurrent neural networks

There has recently been significant interest in RNNs as surrogate models for the computations performed by the brain<sup>13,44–46</sup>. Previous approaches for comparing computations in RNNs for a given task relied on static representations of population activity snapshots<sup>44,47,48</sup>. These methods cannot distinguish dynamics producing similar static representations and require distributional alignment to account for different manifold geometries.

We replicated the delayed-match-to-sample task of<sup>49</sup> by training RNNs with a rank-two connectivity matrix (Fig. 2a, Sect. S3.2), which were previously shown to be sufficiently expressive<sup>50</sup>. The task is commonly used in contextual decision-making and comprises two distinct stimuli with variable



**Figure 2: Statistical representations discriminate computations across recurrent neural networks.** **a** Low-rank RNN, taking two stimuli as input and producing a decision variable as a read-out. **b** Representative stimulation patterns and decision outcomes for the delayed-match-to-sample task for one stimulus. Input amplitude equals the 'gain' during stimulation epochs (red) and zero otherwise (grey). **c** Neural dynamics of a trained rank-two RNN evolves on a randomly oriented plane. The vector field shows mean field dynamics superimposed with a sample trajectory (orange). **d** Input weights to neurons in one RNN solution. The two clusters indicate specialisation in respective inputs. Ellipses represent 3 std of fitted Gaussian distributions. **e** Phase portraits of manifold dynamics for no stimulus and increasing stimulus gains (0.32, 1.0). **f** Distribution distances across gains obtained from geometry-aware embeddings. Hierarchical clustering indicates two clusters in the non-zero gain region, showing a bifurcation (red dashed line). **g** The inferred bifurcation point predicts the drop in task performance by the network. **h** Networks sampled from the coefficient distribution of Solution I have similar dynamics embedded over differently oriented planes, while those trained from other initial conditions have different dynamics. **i** Distribution distances across the three networks and gains obtained from geometry-agnostic embeddings. **j** MDS embedding of the distance matrix in **i** shows alignment of gain-modulated conditions across similar dynamics.



gain and two stimulation epochs of variable duration interspersed by a delay (Fig. 2b). After setting the gain to unity, the RNN network was trained to converge to an output of 1 if either stimulus was presented during both epochs and  $-1$  otherwise (Fig. 2b). As expected<sup>51</sup>, the neural dynamics of the trained network evolve on a randomly oriented plane in neural state-space during a trial (Fig. 2c and Fig. S7a-c). In addition, we found that after training, differently initialised networks produce two classes of solutions characterised by the distributions of their input weights  $\mathbf{w}_1^{\text{in}}, \mathbf{w}_2^{\text{in}}$  (Eq. S22): solution I consist of two subpopulations specialised in sensing the two stimuli (Fig. 2d), whilst in solution II the nodes generalise across the two stimuli (Fig. S7d). These solutions exhibit substantially different neural dynamics as shown by two representative examples, which we selected for further analysis (Fig. 2e, Fig. S7e).

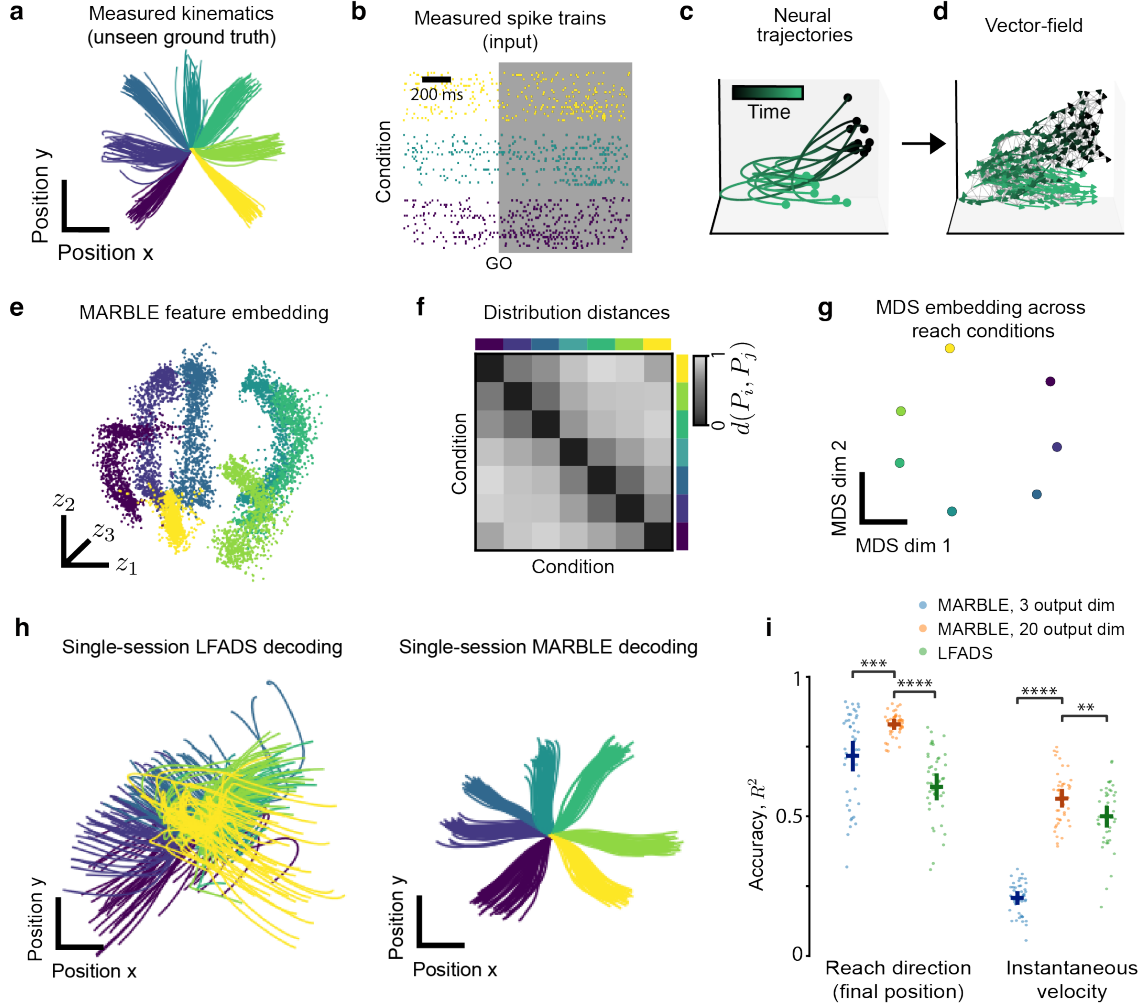
We first asked whether MARBLE could infer continuous and abrupt changes in neural dynamics during varying stimulus gain that predict the loss of task performance. We simulated trajectories with network solution I from 200 randomly initialised trials for 20 different input gains and performed a PCA embedding into three dimensions, explaining over 99% of the variance. Then, we split the trajectories between epochs forming 20 groups of sub-trajectories at no input and 20 groups at different gains, justified by the fact that neural dynamics are the same at a given stimulation gain<sup>11</sup>. We then trained a geometry-aware MARBLE network to jointly embed all groups into a common latent space. The resulting matrix of pairwise distribution distances exhibits a block-diagonal structure (Fig. 2f). The top left block, which corresponds to unstimulated conditions, contains vanishing entries implying that MARBLE is robust to sampling variability. The other two subblocks of the bottom right block indicate an abrupt change in dynamics, which remarkably corresponds to a sudden drop in task performance (Fig. 2g) from 1 to 0.5 (random). This shows that MARBLE enables the unsupervised detection of dynamical events that are interpretable in terms of global decision variables.

We then took both network solution I and II and asked whether comparing distributional embeddings can act as a similarity metric for the dynamics across RNNs (Fig. 2h). In addition to comparing the two trained network solutions, we leveraged the fact that low-rank RNNs sampled from the same Gaussian weight distribution have the same mean field dynamics<sup>50</sup> albeit on differently oriented manifolds. Thus, as a negative control, we generated two networks from the Gaussian distributions fitted to weights of the network solution I (Fig. 2d). We used 200 trials of these three RNNs to produce feature embeddings for stimulation epochs with a geometry-agnostic MARBLE network. We found a block diagonal structure in the distribution distance matrix across gain-modulated conditions which clearly discriminates between the on-manifold dynamics of networks sampled from Solution I and II (Fig. 2i). In addition to correctly clustering solutions, the MDS embedding of the distance matrix (Fig. 2j) also shows that MARBLE reveals the underlying one-parameter variation due to gain modulation, which is correctly ordered within a network and across sampled networks. These findings confirm that our method can be used to obtain a robust metric between dynamical processes generated by different system architectures with possible implications towards guiding their design for faster training or more accurate predictions.

## Representing and decoding neural dynamics in a macaque reaching task

To show that distributional representations generated by MABRLE can be used to interpret neural dynamics in terms of behavioural variables, we reanalysed the electrophysiological recordings of a macaque performing a delayed centre-out hand-reaching task<sup>24</sup> (Sect. S3.3. During the task, a trained monkey was instructed to move a handle towards seven distinct targets placed at radial locations from the start position. The dataset comprised simultaneous recordings of hand kinematics (Fig. 3a) and neural activity via a 24-channel probe inserted into the premotor cortex (PMd) over 44 recording sessions (Fig. 3b shows one session).

Since a subset of neurons in PMd is directionally tuned<sup>52</sup>, we argued that neural representations must be sensitive to the orientation of the neural manifold. Thus, we trained a geometry-aware MARBLE network on individual sessions by constructing separate vector fields from the firing rate traces under individual reach conditions (Fig. 3c-d). Doing so avoids imposing geometric relationships between conditions and allowed us to test if behaviourally interpretable embeddings emerge from the neural dynamics. Unexpectedly, we found that, unlike the popular latent factor encoding of neural dynamics (Fig. S8a), the raw MARBLE feature embeddings in 3D strongly reflect the spatial geometry of the physical reaches (Fig. 3e and Fig. S8a). This geometric configuration is also confirmed by the diagonal and periodic structure of the condition-averaged distance matrix between feature



**Figure 3: Interpretable representation and decoding of neural activity in a macaque centre-out reaching task.** **a** Ground truth hand trajectories in seven reaching directions. **b** Single-trial spike-trains in premotor cortex during a single trial for three respective reach directions (24 recording channels). Shaded area shows the analysed traces after the GO cue. **c** Ten representative firing rate trajectories PCA-embedded in 3D for visualisation. **d** Example vector field obtained from firing rate trajectories for a given reach condition. **e** Neural data from all conditions in a single session are used to train a MARBLE network without information about the kinematics. Statistical representation from MARBLE of neural dynamics for a given session geometrically corresponds to the reach configuration in physical space. **f** Matrix of distribution distances across reach conditions averaged across sessions shows periodic structure. **g** MDS embedding of the distance matrix recovers the cyclic ordering of reaches. **h** Hand trajectories linearly decoded from a representative session from latent factors of the best LFADS model<sup>24</sup> and from MARBLE embeddings. **i** Decoding accuracy measured by  $R^2$  between ground truth and decoded trajectories across all sessions for final position (left) and instantaneous velocity (right). Wilcoxon tests (paired samples), \*\*:  $p < 1 \times 10^{-2}$ , \*\*\*:  $p < 1 \times 10^{-3}$ , \*\*\*\*:  $p < 1 \times 10^{-4}$ .

distributions across reach directions (Fig. 3f, Fig. S8b) and its associated MDS embedding (Fig. 3g, Fig. S8c). This spatial structure was lost when training a geometry-agnostic network, which further evidence the role of neural manifold geometry in coding reach movements (Fig. S8). Hence, MARBLE can unfold the global geometric information in the neural code that mirrors kinematics in physical space.

This geometric correspondence between neural and behavioural representations suggests a novel decoding algorithm. Departing from current decoders which rely on explicitly learning temporal information<sup>23–25</sup> here we asked whether temporal information could emerge naturally from feature embeddings due to the spatial continuity of local vector fields over the manifold. To show this, we fitted an optimal linear estimator between the MARBLE embedded time points and their corresponding hand positions. Remarkably, this decoded kinematics outperformed the state-of-the-art LFADS model<sup>24</sup> (Fig. 3h) as quantified by the higher accuracy in the reach direction (Fig. 3i) for all output dimensions. While a three-dimensional MARBLE embedding accurately encodes spatial positions, we found that the delay between the GO cue and the beginning of the movement could only be accounted for with higher embedding dimensions (Fig. 3i). These results highlight that MARBLE can generate representations of neural dynamics that are simultaneously interpretable and decodable into behavioural variables.

## Discussion

A hallmark of large collective systems such as the brain is the existence of many system realisations that lead to equivalent computations defined by population-level dynamical processes<sup>53,54</sup>. The growing recognition that dynamics in biological and artificial neural networks evolve over low-dimensional manifolds<sup>5,6,8,9</sup> offers an opportunity to reconcile variable dynamics across system realisations with invariant computations by using manifold geometry as an inductive bias for designing data-driven models. We have shown that non-linear dynamical systems can be represented as statistical distributions built from the point-wise, similarity-preserving embedding of local vector fields. Due to the spatial continuity of dynamics over the manifold, latent features can be learnt fully unsupervised using a geometric deep learning architecture. Further, features can be made rotation invariant to achieve representations that are robust to perturbations in manifold geometry and orientation. These properties open the door to independently testing the relevance of manifold geometry and dynamics, and to comparing dynamics across system realisations.

Our formalism can be framed as a statistical generalisation of the convergent cross mapping (CCM) framework by Sugihara et al.<sup>29</sup>, which tests the topological equivalence of two dynamical attractors through a one-to-one map between their local neighbourhoods. While CCM tests the causality between two long, concurrent time series, our method provides a similarity metric between any collection of dynamical systems based on ensembles of variable-length, sparsely sampled time series. In addition, due to the locality of representation, our approach diverges from typical geometric deep learning models that learn vector fields globally<sup>35,40</sup>, and thus unable to consider the manifold geometry and the dynamics separately. Locality also implies that our method can generalise to assimilate different datasets without additional trainable parameters to increase the statistical power of the model even when individual datasets are poorly sampled. Although our method does not explicitly learn temporal dynamics over manifolds<sup>23,37</sup>, we showed that temporal ordering naturally emerges from our similarity-preserving embeddings of local vector fields. We demonstrated that this property enables kinematic decoding from embeddings of neural data. Beyond decoding, we expect MARBLE embeddings to provide powerful representations for downstream machine-learning tasks such as prediction or classification.

Despite the variety of representations of a dynamical system depending on applications, we provide here a general, model-free approach to be used as an unbiased first-line tool for studying the dynamical effects of experimental interventions before applying specific model-based methods. The unique ability of our approach to learning global properties of dynamics from local vector fields points towards building predictive models with modern transformer architectures using local vector fields as tokens. Finally, constraining dynamical data over a manifold regularises the space of possible dynamics, which can in turn be used as a test of geometric consistency for scientific hypotheses and model validation across disciplines.



## Code availability

The code to carry out the simulations and analysis can be found at [github.com/agosztolai/MARBLE](https://github.com/agosztolai/MARBLE).

## Data availability

The data generated during the simulations is available with DOI: [10.7910/DVN/KTE4PC](https://doi.org/10.7910/DVN/KTE4PC).

## Acknowledgements

We thank Nicolas Aspert for the much-needed computing support. We also thank Adrian Valente, Nikolas Karalias and Matteo Vинаo-Carl for the interesting discussions. AG acknowledges support from an HFSP Cross-disciplinary Postdoctoral Fellowship (LT000669/2020-C). MB acknowledges funding through EPSRC awards EP/N014529/1 (Centre for Mathematics of Precision Healthcare) and EP/W024020/1 (Statistical Physics of Cognition). RP acknowledges the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Project-ID 424778381-TRR 295.

## Author contributions

Conceptualisation: A.G., R.L.P., A.A., M.B., P.V., Methodology: A.G., R.L.P., A.A., P.V., Software: A.G., R.L.P., A.A., Analysis: A.G., R.L.P., A.A., Writing - Original Draft: A.G., R.L.P., A.A., Writing - Review & Editing: M.B., P.V.

## Competing interests

The authors declare no competing interests.

## References

1. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U. S. A.* **79**, 2554–2558 (1982).
2. Churchland, M. M. *et al.* Neural population dynamics during reaching. *Nature* **487**, 51–56 (2012).
3. Duncker, L. & Sahani, M. Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Curr. Opin. Neurobiol.* **70**, 163–170 (2021).
4. Gallego, J. A. *et al.* Neural manifolds for the control of movement. *Neuron* **94**, 978–984 (2017).
5. Chung, S., Lee, D. D. & Sompolinsky, H. Classification and geometry of general perceptual manifolds. *Phys. Rev. X* **8**, 31003 (2018).
6. Chaudhuri, R. *et al.* The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nat. Neurosci.* **22**, 1512–1520 (2019).
7. Kriegeskorte, N. & Wei, X.-X. Neural tuning and representational geometry. *Nat. Rev. Neurosci.* **22**, 703–718 (2021).
8. Khona, M. & Fiete, I. R. Attractor and integrator networks in the brain. *Nat. Rev. Neurosci.* **23**, 744–766 (2022).
9. Gardner, R. J. *et al.* Toroidal topology of population activity in grid cells. *Nature* **602**, 123–128 (2022).
10. Nogueira, R. *et al.* The geometry of cortical representations of touch in rodents. *Nat. Neurosci.* **26**, 239–250 (2023).
11. Beiran, M. *et al.* Parametric control of flexible timing through low-dimensional neural manifolds. *Neuron* **111**, 739–753.e8 (2023).

12. Langdon, C., Genkin, M. & Engel, T. A. A unifying perspective on neural manifolds and circuits for cognition. *Nat. Rev. Neurosci.* (2023).
13. Sussillo, D. & Barak, O. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput.* **25**, 626–649 (2013).
14. Toi, P. T. *et al.* In vivo direct imaging of neuronal activity at high temporospatial resolution. *Science* **378**, 160–168 (2022).
15. Steinmetz, N. A. *et al.* Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science* **372**, eabf4588 (2021).
16. Vilette, V. *et al.* Ultrafast Two-Photon Imaging of a High-Gain Voltage Indicator in Awake Behaving Mice. *Cell* **179**, 1590–1608.e23 (2019).
17. Lauer, J. *et al.* Multi-animal pose estimation, identification and tracking with DeepLabCut. *Nat. Methods* **19**, 496–504 (2022).
18. Pereira, T. D. *et al.* SLEAP: A deep learning system for multi-animal pose tracking. *Nat. Methods* **19**, 486–495 (2022).
19. Gosztolai, A. *et al.* LiftPose3D, a deep learning-based approach for transforming two-dimensional poses to three-dimensional poses in laboratory animals. *Nat. Methods* **18**, 975–981 (2021).
20. Brunton, S. L. *et al.* Chaos as an intermittently forced linear system. *Nat. Commun.* **8**, 1–8 (2017).
21. Lusch, B., Kutz, J. N. & Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.* **9**, 851–10 (2018).
22. Kipf, T. *et al.* Neural Relational Inference for Interacting Systems. *arXiv preprint arXiv:1802.04687* (2018).
23. Low, R. J. *et al.* Probing variability in a cognitive map using manifold inference from neural dynamics. *bioRxiv* (2018).
24. Pandarinath, C. *et al.* Inferring single-trial neural population dynamics using sequential auto-encoders. *Nat. Methods*, 1–21 (2018).
25. Zhu, F. *et al.* A deep learning framework for inference of single-trial neural population dynamics from calcium imaging with subframe temporal resolution. *Nat. Neurosci.* **25**, 1724–1734 (2022).
26. Yair, O. *et al.* Reconstruction of normal forms by learning informed observation geometries from data. *Proc. Natl. Acad. Sci. U. S. A.* **114**, E7865–E7874 (2017).
27. Dabagia, M. & Konrad, P. Comparing high-dimensional neural recordings by aligning their low-dimensional latent representations. *Nat. Biomed. Eng.*, 1–16 (2021).
28. Schneider, S., Lee, J. H. & Mathis, M. W. Learnable latent embeddings for joint behavioral and neural analysis. *CoRR* **abs/2204.00673** (2022).
29. Sugihara, G. *et al.* Detecting causality in complex ecosystems. *Science* (80-. ). **338**, 496–500 (2012).
30. Gosztolai, A. & Arnaudon, A. Unfolding the multiscale structure of networks with dynamical Ollivier-Ricci curvature. *Nat. Commun.* **12**, 4561 (2021).
31. Skinner, D. J. *et al.* Topological metric detects hidden order in disordered media. *Phys. Rev. Lett.* **126**, 48101 (2021).
32. Bronstein, M. M. *et al.* Geometric Deep Learning: going beyond euclidean data. *IEEE Signal Process. Mag.* **34**, 18–42 (2017).
33. Masci, J. *et al.* Geodesic Convolutional Neural Networks on Riemannian Manifolds in 2015 IEEE International Conference on Computer Vision Workshop (ICCVW) (IEEE Computer Society, Los Alamitos, CA, USA, 2015), 832–840.
34. Bronstein, M. M. *et al.* Geometric Deep Learning: grids, groups, graphs, geodesics, and gauges 2021.
35. Grattarola, D. & Vandergheynst, P. Generalised implicit neural representations. *Advances in Neural Information Processing Systems* (2022).

36. Jenner, E. & Weiler, M. *Steerable Partial Differential Operators for Equivariant Neural Networks* in *International Conference on Learning Representations* (2022).
37. Floryan, D. & Graham, M. D. Data-driven discovery of intrinsic dynamics. *Nat. Mach. Intell.* **4**, 1113–1120 (2022).
38. Sharp, N. *et al.* DiffusionNet : discretization agnostic learning on surfaces. *ACM Trans. Graph.* **99** (2020).
39. Beaini, D. *et al.* *Directional graph networks* in *International Conference on Machine Learning* (2021), 748–758.
40. Bodnar, C. *et al.* *Neural Sheaf Diffusion: A topological perspective on heterophily and over-smoothing in GNNs* in *Advances in Neural Information Processing Systems* (eds Oh, A. H. *et al.*) (2022).
41. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017-Decem**, 1025–1035 (2017).
42. Fefferman, C., Mitter, S. & Narayanan, H. Testing the manifold hypothesis. *Journal of the American Mathematical Society* **29**, 983–1049 (2016).
43. McInnes, L. *et al.* UMAP: Uniform Manifold Approximation and Projection. *The Journal of Open Source Software* **3**, 861 (2018).
44. Flesch, T. *et al.* Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron* **110**, 1258–1270.e11 (2022).
45. Rajalingham, R., Piccato, A. & Jazayeri, M. Recurrent neural networks with explicit representation of dynamic latent variables can mimic behavioral patterns in a physical inference task. *Nat. Commun.* **13** (2022).
46. Galgali, A. R., Sahani, M. & Mante, V. Residual dynamics resolves recurrent contributions to neural computation. *Nat. Neurosci.* (2023).
47. Kornblith, S. *et al.* Similarity of neural network representations revisited. *36th Int. Conf. Mach. Learn. ICML 2019* **2019-June**, 6156–6175 (2019).
48. Williams, A. H. *et al.* Generalized shape metrics on neural representations. *Adv. Neural Inf. Process. Syst.* **6**, 4738–4750 (2021).
49. Miyashita, Y. Neuronal correlate of visual associative long-term memory in the primate temporal cortex. *Nature* **335**, 817–820 (1988).
50. Dubreuil, A. *et al.* The role of population structure in computations through neural dynamics. *Nat. Neurosci.* **25**, 783–794 (2022).
51. Mastrogiuseppe, F. & Ostojic, S. Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks. *Neuron* **99**, 609–623.e29 (2018).
52. Riehle, A. & Requin, J. Monkey primary motor and premotor cortex: single-cell activity related to prior information about direction and extent of an intended movement. *Journal of Neurophysiology* **61**, 534–549 (1989).
53. Marder, E. & Taylor, A. L. Multiple models to capture the variability in biological neurons and networks. *Nat. Neurosci.* **14**, 133–138 (2011).
54. Gosztolai, A. & Ramdya, P. Connecting the dots in ethology: applying network theory to understand neural and animal collectives. *Curr. Opin. Neurobiol.* **73**, 102532 (2022).
55. Mount, D. M. & Arya, S. Finding the nearest neighbors for points in d-dimensional Euclidean space. *ACM Transactions on Mathematical Software (TOMS)* **24**, 96–103 (1998).
56. Berry, T. & Sauer, T. Consistent manifold representation for topological data analysis. *Foundations of Data Science* **1**, 1–38 (2019).
57. Budninskiy, M. *et al.* Parallel Transport Unfolding: A Connection-Based Manifold Learning Approach. *SIAM Journal on Applied Algebra and Geometry* **3**, 266–291 (2019).
58. Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A* **32**, 922–923 (Sept. 1976).
59. Berline N. Getzler E., V. M. *Heat kernels and Dirac operators* 2nd. (Springer, 1996).

- 60. Singer, A. & Wu, H. T. Vector diffusion maps and the connection Laplacian. *Commun. Pure Appl. Math.* **65**, 1067–1144 (2012).
- 61. He, K. *et al.* Deep Residual Learning for Image Recognition in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016), 770–778.
- 62. Fey, M. & Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric in ICLR Workshop on Representation Learning on Graphs and Manifolds (2019).
- 63. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization 2014.
- 64. Denker, M., Yegenoglu, A. & Grün, S. Collaborative HPC-enabled workflows on the HBP Collaboratory using the Elephant framework in Neuroinformatics 2018 (2018), P19.

# Supplementary Material for: Interpretable statistical representations of neural population dynamics and geometry

Adam Gosztolai,<sup>1\*</sup> Robert L. Peach,<sup>2,3</sup> Alexis Arnaudon,<sup>4</sup>  
Mauricio Barahona,<sup>4</sup> Pierre Vanderghelynst<sup>1</sup>

<sup>1</sup>Signal Processing Laboratory (LTS2), EPFL, Lausanne, 1016, Switzerland

<sup>2</sup>Department of Neurology, University Hospital Würzburg, Würzburg, 97070, Germany

<sup>3</sup>Department of Brain Sciences, Imperial College London, London, W12 0NN, United Kingdom,

<sup>4</sup>Department of Mathematics, Imperial College London, London, SW7 2AZ, United Kingdom

\*To whom correspondence should be addressed; E-mail: adam.gosztolai@epfl.ch.

## S1 The MARBLE method

The MARBLE method aims to find a similarity-preserving map from the local vector fields to a shared latent space. Since the method processes vector fields point-by-point, it suffices to describe it applied to a single vector field and the generalisation to the joint embedding of multiple vector fields is straightforward.

Given a smooth, compact  $m$ -dimensional manifold  $\mathcal{M}$  described by a point cloud  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ , MARBLE aims at representing a set of vector fields  $\mathbf{F} = \{\mathbf{f}_1, \dots, \mathbf{f}_n\}$  on this manifold as an empirical distribution  $P(\mathbf{Z}) = \sum_{i=1}^n \delta(\mathbf{z}_i)$  on a latent space with coordinates  $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_n)$ . Such vector fields can be obtained from time-series data with for example first-order finite difference  $\mathbf{f}_i := \mathbf{x}(t_i+1) - \mathbf{x}(t_i)$ . For a chosen set of points on this manifold, we obtain their latent space coordinates  $\mathbf{Z}$  from the vector field restricted to a neighbourhood around each point with an unsupervised learning architecture.

The resulting statistical distribution  $P_c$  on the latent space then represents reduced information about the underlying dynamical system. If subsets of points belonging to different dynamical systems are considered together, the embedding can be used to disentangle differences in their dynamics, while being invariant, or not to the geometry encoded in the manifold.

### S1.1 Approximating the manifold by a proximity graph

To define the local neighbourhood on  $\mathcal{M}$ , we describe it with a proximity graph. To obtain a faithful representation of the tangent space of  $\mathcal{M}$  via the finite difference vector fields  $\mathbf{F}$ , we need to ensure that the graph is as homogeneous as possible. In particular, if points come from dynamical trajectories, taking consecutive temporal points should be prohibited. For this reason, we generate the proximity graph from a subsample of points obtained by farthest point sampling<sup>55</sup>, with a criterion  $\beta \in [0, 1]$  that controls the spacing of the points relative to the diameter of the manifold  $\max_{i,j} (\|\mathbf{x}_i - \mathbf{x}_j\|_2) < \beta \text{diam}(\mathcal{M})$ .

Then, several classical proximity graph algorithms can be used, such as the  $k$ -NN algorithm and the  $\epsilon$ -ball algorithm, but we found that the continuous  $k$ -nearest neighbour ( $ck$ -NN) algorithm<sup>56</sup> creates more representative neighbourhoods. Indeed, contrary to the classical  $k$ -NN graph algorithm, it can be interpreted as a local kernel density estimate and accounts for sampling density variations over  $\mathcal{M}$ . The  $ck$ -NN algorithm extends the classical  $k$ -NN algorithm by accounting for density variations by connecting  $i$  and  $j$  whenever  $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 < \delta \|\mathbf{x}_i - \mathbf{x}_u\|_2 \|\mathbf{x}_j - \mathbf{x}_v\|_2$ , where  $\delta > 0$  is a scaling parameter,  $u, v$  are the  $k$ -th nearest neighbours of  $i, j$ , respectively, and  $\|\cdot\|_2$  is the Euclidean



norm. This proximity graph endows  $\mathcal{M}$  with a geodesic structure, i.e., for any  $i, j \in \mathcal{M}$  there is a shortest path with distance  $d(i, j)$ . We can then define the local vector fields as the  $p$ -hop geodesic neighbourhood  $\mathcal{N}(i, p)$  around each point  $i$ .

### S1.2 Parametrising the tangent spaces

Unlike  $\mathbb{R}^m$ , the tangent spaces of  $\mathcal{M}$  do not come with a preferred coordinate system. However, being isomorphic to  $\mathbb{R}^m$ , they can be parametrised by a family of local orthogonal coordinate frames, also known as gauges, whose components are  $\mathbb{R}^d$ -valued vectors. Specifically, we assume that the tangent space at a node  $i$ ,  $\mathcal{T}_i\mathcal{M}$ , is spanned by the edge vectors  $\mathbf{e}_{ij} \in \mathbb{R}^d$  pointing from  $i$  to  $K$  nodes  $j$  in its neighbourhood on the proximity graph. The  $m$  largest singular values  $\mathbf{t}_i^{(\cdot)} \in \mathbb{R}^d$  of the matrix formed by column-stacking  $\mathbf{e}_{ij}$  yield the orthonormal coordinate frame  $\mathbb{T}_i \in \mathbb{R}^{d \times m} = (\mathbf{t}_i^{(1)}, \dots, \mathbf{t}_i^{(m)})$  spanning  $\mathcal{T}_i\mathcal{M}$ . In practice, we pick  $K > \deg(i)$  closest nodes to  $i$  on the proximity graph where  $K$  is a hyperparameter. Larger  $K$  increases the overlaps between the nearby tangent spaces and we find that  $K = 2|\mathcal{N}(i, 1)|$  is often a good compromise between locality and robustness to noise of the tangent space approximation. Note that because  $\mathbb{T}_i$  defines an orthogonal basis,  $\mathbb{T}_i^T \mathbf{f}_i$  acts as a projection of the signal to the tangent space in the  $\ell_2$  sense. We perform these computations using a modified version of the Parallel Transport Unfolding package<sup>57</sup>. For illustration, we display the computed frames on a spherical manifold (Fig. S1).

### S1.3 Connections between tangent spaces

Having the local frames, we next define the parallel transport map  $\mathcal{P}_{j \rightarrow i}$  aligning the local frame at  $j$  to that at  $i$ , which is necessary to define convolution operations in a common space. While parallel transport is generally path dependent, we assume that adjacent nodes  $i, j$  are close enough to consider the unique smallest rotation, known as the Lévy-Civita connection. Thus, for adjacent edges,  $\mathcal{P}_{j \rightarrow i}$  can be computed as the matrix  $\mathbf{O}_{ji}$  corresponding to  $\mathcal{P}_{j \rightarrow i}$ , as the orthogonal transformation (rotation and reflection)

$$\mathbf{O}_{ji} = \arg \min_{\mathbf{O} \in O(m)} \|\mathbb{T}_i - \mathbb{T}_j \mathbf{O}\|_F, \quad (\text{S1})$$

where  $\|\cdot\|_F$  is the Frobenius norm. The unique solution to this problem is found by the Kabsch algorithm<sup>58</sup>.

Note that  $\mathbb{T}_i$  is defined only up to an orthogonal transformation (rotation and reflection) within the tangent space of  $\mathcal{M}$  because the  $m$ -dimensional  $\mathcal{T}_i\mathcal{M}$  only constrains  $m$  coordinates of the frame. However, when the signal is projected to the local frame, the tangent frame alignment by  $\mathcal{P}_{j \rightarrow i} = \mathbf{O}_{ij}$  removes this ambiguity. Indeed, suppose that each node carries the same signal  $\mathbf{f}$ , then, parallel transport alignment of the projected signal from  $j$  to  $i$  yields

$$\mathbb{T}_i^T \mathbf{f} = \mathbf{O}_{ij} \mathbb{T}_j^T \mathbf{f} = (\mathbb{T}_j \mathbf{O}_{ji})^T \mathbf{f} = \mathbb{T}_i^T \mathbf{f}, \quad (\text{S2})$$

where the first equality used the definition of the parallel transport, the second equality the transpose operation, while the third equality used Eq. S1. Note that the same result does not hold when parallel transporting signals in the ambient space (without projection), because in that case, the ambiguity in the frame orientation introduces ambiguity in the signal

### S1.4 Vector diffusion

Before constructing the vector field features, we use a vector diffusion layer to smooth out noisy vector field samples. Vector diffusion is a generalisation of the scalar (heat) diffusion, and can be expressed as a kernel associated with the vector diffusion equation<sup>59</sup> to produce a smoothed vector field

$$\text{vec}(\mathbf{F}(\tau)) = e^{-\tau \mathcal{L}} \text{vec}(\mathbf{F}), \quad (\text{S3})$$

where  $\text{vec}(\mathbf{F}) \in \mathbb{R}^{nd \times 1}$  the row-wise concatenation of vector-valued signals,  $\tau$  is a learnable parameter that controls the scale of the local vector fields and  $\mathcal{L}$  is the random-walk normalised connection Laplacian defined as a block matrix whose nonzero blocks are given by

$$\mathcal{L}(i, j) = \begin{cases} \mathbf{I}_{m \times m} & \text{for } i = j \\ -\deg(i)^{-1} \mathbf{O}_{ij} & \text{for } j \in \mathcal{N}(i, 1). \end{cases} \quad (\text{S4})$$

See<sup>60</sup> for further details. Intuitively, rather than diffusion of the vectors component-wise, vector diffusion smoothens vectors based on differences between a vector at a given node and other vectors parallel transported to this node.

### S1.5 Gradient filters

Before we can learn the vector field features, we define gradient filters, whose role is to approximate the variation of the vector field around a point in the local tangent frame. To numerically compute the gradient at a point  $i \in \mathcal{M}$  we construct an anisotropic filter by extending the directional derivative filter of<sup>39</sup> using local coordinates. To do so, consider the local frame  $\mathbb{T}_i$  and construct the directional derivative filter in the direction of the  $q$ -th unit vector  $\mathbf{t}_i^{(q)}$ . We follow<sup>39</sup> and decompose  $\mathbf{t}_i^{(q)} \in \mathbb{R}^{d \times 1}$  by projecting it to the set of edge vectors  $\mathbf{e}_{ij}$  to obtain a vector  $\hat{\mathbf{t}}_i^{(q)} \in \mathbb{R}^{n \times 1}$  at node  $i$  pointing in the basis direction  $q$  with components  $j$

$$\hat{t}_i^{(q)}(j) = \begin{cases} \langle \mathbf{t}_i^{(q)}, \mathbf{e}_{ij} \rangle / \deg(i) & \text{if } j \in \mathcal{N}(i, 1) \\ 0 & \text{otherwise.} \end{cases} \quad (\text{S5})$$

Collating for all nodes, we can write a matrix for the  $q$ -th coordinate of the local frame projected onto the edge vectors  $\hat{\mathbb{T}}_q = (\hat{\mathbf{t}}_1^{(q)}, \dots, \hat{\mathbf{t}}_n^{(q)}) \in \mathbb{R}^{n \times n}$ . Using this decomposition, the directional derivative of the scalar field  $s_i$  at  $i$  in the direction  $\mathbf{t}_i^q$  is given by

$$\mathcal{K}^{(i,q)} s_i := \langle \nabla s_i, \hat{\mathbf{t}}_i^{(q)} \rangle = \sum_{j \in \mathcal{N}(i,1)} (s_j - s_i) \hat{t}_i^{(q)}(j), \quad (\text{S6})$$

or, in matrix form, by

$$\mathbf{K}^{(q)} \mathbf{s} = (\hat{\mathbb{T}}_q - \text{diag}(\hat{\mathbb{T}}_q \mathbf{1}_n)) \mathbf{s}, \quad (\text{S7})$$

where  $\mathbf{1}_n$  is the  $n \times 1$  vector of ones. As a result, the gradient of a scalar field can be obtained by column-wise concatenating (as new channels) the derivatives against all directions in the basis set

$$\nabla \mathbf{s} = (\mathcal{D}^{(1)} \mathbf{s}, \dots, \mathcal{D}^{(d)} \mathbf{s}). \quad (\text{S8})$$

Figs. S2a,b show the output of the first and second-order filters applied to a linear and a quadratic scalar field.

To compute the component-wise directional derivative for a vector field  $\mathbf{F} \in \mathbb{R}^{n \times m}$ , one must first parallel transport the local frames at the neighbours  $j$  to  $i$  before applying the anisotropic filters in Eq. S7. Let  $\mathbf{O}$  denote the  $nm \times nm$  block matrix of  $m \times m$  blocks given by the connection matrices  $\mathbf{O}_{ij}$ . Then, we may express Eq. S12 in matrix form as

$$\mathcal{D}^{(q)} \mathbf{F} = ((\mathbf{K}^{(q)} \otimes \mathbf{1}_m^T \mathbf{1}_m) \odot \mathbf{O}) \mathbf{F}. \quad (\text{S9})$$

Here the kronecker product in the inner brackets expands  $\mathbf{K}^{(q)}$  to the  $nm \times nm$  block matrix where the  $(i, j)$   $m \times m$  block is filled with entries  $K_{ij}^{(q)}$ .

### S1.6 Approximating local vector fields

We now define convolution kernels on  $\mathcal{M}$  that act on the vector field to give a representation of the vector field around a given point. We first project the vector signal to the manifold  $\mathbf{f}'_i = \mathbb{T}_i^T \mathbf{f}_i$ . This reduces the dimension of  $\mathbf{f}_i$  from  $d$  to  $m$  without losing information since  $\mathbf{f}_i$  was already in the tangent space. We drop the bar in the sequel to understand that all vectors are expressed in local coordinates. In this local frame, the best polynomial approximation of the vector field around  $i$  is given by the Taylor-series expansion of each component  $f_{i,l}$  of  $\mathbf{f}_i$

$$f_{j,l} \approx f_{i,l} + \nabla f_{i,l} (\mathbf{x}_j - \mathbf{x}_i) + \frac{1}{2} (\mathbf{x}_j - \mathbf{x}_i)^T \nabla^2 f_{i,l} (\mathbf{x}_j - \mathbf{x}_i) + \dots \quad (\text{S10})$$

Motivated by the Taylor approximation, we construct gradient operators of increasing order and implement them as a set of  $m$  anisotropic filters  $\{\mathcal{D}^{(q)}\}$  acting along  $\{\mathbf{t}_i^{(q)}\}$ ,

$$\nabla f_{i,l} \approx \left( \mathcal{D}^{(1)}(f_{i,l}), \dots, \mathcal{D}^{(m)}(f_{i,l}) \right)^T. \quad (\text{S11})$$

Here,  $\mathcal{D}^{(q)}(f_{i,l})$  is the  $l$ -th component of

$$\mathcal{D}^{(q)}(\mathbf{f}_i) = \sum_{j=1}^n \mathcal{K}_j^{(i,q)} \mathcal{P}_{j \rightarrow i}(\mathbf{f}_j), \quad (\text{S12})$$

where  $\mathcal{P}_{j \rightarrow i} = \mathbf{O}_{ij}$  is the parallel transport operator that takes the vector  $\mathbf{f}_j$  from the adjacent frame  $j$  to a common frame at  $i$ .  $\mathcal{K}^{(i,q)} \in \mathbb{R}^{n \times n}$  is a directional derivative filter<sup>39</sup> (Eq. S7) expressed in local coordinates at  $i$  and acting along  $\mathbf{t}_i^{(q)}$ . As a result of the parallel transport, the value of Eq. S12 is independent of the local curvature of the manifold. The  $p$ -th order gradients can be defined by the iterated application of the filters, which aggregates information in the  $p$ -hop neighbourhood of points. Although we find that increasing the order of the differential operators increases the expressiveness of the network (Fig. S3), second-order filters ( $p = 2$ ) were sufficient for the application considered in this paper.

The expansion in Eq. S10 suggests augmenting the vectors  $\mathbf{f}_i$  by the derivatives (Eq. S11), to obtain a matrix

$$\mathbf{f}_i \mapsto \mathbf{f}_i^{\mathcal{D}} = (\mathbf{f}_i, \nabla f_{i,1}, \dots, \nabla f_{i,m}, \nabla(\nabla f_{i,1})_1, \dots, \nabla(\nabla f_{i,m})_m), \quad (\text{S13})$$

of dimensions  $m \times c$  whose columns are gradients of signal components up to order  $p$  to give a total of  $c = (1 - m^{p+1})/(m(1 - m))$  vectorial channels.

### S1.7 Inner product for geometry invariance

In the optional geometry-agnostic mode, deformations on the manifold have the effect of introducing rotations into the local vector fields. Thus, we can achieve invariance to these deformations by making the learnt features rotation invariant. We do so by first transforming the  $m \times c$  matrix  $\mathbf{f}_i^{\mathcal{D}}$  to a  $1 \times c$  vector as

$$\mathbf{f}_i^{\mathcal{D}} \mapsto \mathbf{f}_i^{\text{ip}} = \left( \mathcal{E}^{(1)}(\mathbf{f}_i^{\mathcal{D}}), \dots, \mathcal{E}^{(c)}(\mathbf{f}_i^{\mathcal{D}}) \right). \quad (\text{S14})$$

Then, by taking for each channel the inner product against all other channels, weighted by a dense learnable matrix  $\mathbf{A}^{(r)} \in \mathbb{R}^{m \times m}$  and summing, we obtain

$$\mathcal{E}^{(r)}(\mathbf{f}_i^{\mathcal{D}}) = \mathcal{E}^{(r)}(\mathbf{f}_i^{\mathcal{D}}; \mathbf{A}^{(r)}) := \sum_{s=1}^c \left\langle \mathbf{f}_i^{\mathcal{D}}(\cdot, r), \mathbf{A}^{(r)} \mathbf{f}_i^{\mathcal{D}}(\cdot, s) \right\rangle, \quad (\text{S15})$$

for  $r = 1, \dots, c$  (Fig. 1f). Taking inner products is valid because the columns of  $\mathbf{f}_i^{\mathcal{D}}$  all live in the tangent space at  $i$ . Intuitively, Eq. S15 achieves coordinate independence by learning rotation and scaling relationships between pairs of channels.

### S1.8 Embedding with a multilayer perceptron (MLP)

To embed each local feature,  $\mathbf{f}_i^{\text{ip}}$  or  $\mathbf{f}_i^{\mathcal{D}}$ , depending on if inner product features are used, (Eq. S14) we use a multilayer perceptron (MLP) (Fig. 1g)

$$\mathbf{z}_i = \text{MLP}(\mathbf{f}_i^{\text{ip}}; \omega), \quad (\text{S16})$$

where  $\omega$  are trainable weights. The MLP is composed of  $L$  linear (fully-connected) layers interspersed by ReLU non-linearities. We used  $L = 2$  with a sufficiently high output dimension to encode the variables of interest. The parameters were initialised using the Kaiming method<sup>61</sup>.

### S1.9 Loss function

Unsupervised training of the network is possible due to the continuity in the vector field over  $\mathcal{M}$ , which causes nearby local vector fields to be more similar than distant ones. We implement this via negative sampling<sup>41</sup>, which uses random walks sampled at each node to embed neighbouring points on the manifold close together while pushing points sampled uniformly at random far away. We use the following unsupervised loss function<sup>41</sup>

$$\mathcal{J}(\mathbf{Z}) = -\log(\sigma(\mathbf{z}_i^T \mathbf{z}_j)) - Q \mathbb{E}_{k \sim U(n)} \log(\sigma(-\mathbf{z}_i^T \mathbf{z}_k)), \quad (\text{S17})$$

where  $\sigma(x) = (1 + e^{-x})^{-1}$  is the sigmoid function and  $U(n)$  is the uniform distribution over the  $n$  nodes. To compute this function, we sample one-step random walks from every node  $i$  to obtain 'positive' node samples for which we expect similar local vector fields to that of node  $i$ . The first term in Eq. S17 seeks to embed these nodes close together. At the same time, we also sample nodes uniformly at random to obtain 'negative' node samples with likely different local vector fields from that of node  $i$ . The second term in Eq. S17 seeks to embed these nodes far away. We also choose  $Q = 1$ .

We optimise the loss Eq. S17 by stochastic gradient descent. For training, the nodes from all manifolds were randomly split into training (80%), validation (10%) and test (10%) sets. The optimiser was run until convergence of the validation set and the final results were tested on the test set with the optimised parameters.

### S1.10 Pseudo code of MARBLE algorithm

We implemented MARBLE architecture with Pytorch Geometric<sup>62</sup>. The general algorithm is as follows.

---

#### Algorithm 1 MARBLE

---

**Input:**  $d$ -dimensional vector field samples  $\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_n)$   
connection Laplacian  $\mathcal{L}$   
derivative filters  $\mathcal{D}_i^{(q)}$  for  $i \in \{1, \dots, n\}$  and  $q \in \{1, \dots, m\}$   
derivative order  $p$

**Output:** Embedding  $\mathbf{z}_i$  for all  $i \in \{1, \dots, n\}$

```

 $\mathbf{F} \leftarrow e^{\tau \mathcal{L}} \mathbf{F}$  ▷ Apply diffusion layer (optional)
 $\mathbf{h}^{(0)} \leftarrow \mathbf{f}_i$ 
for  $1 \leq l \leq p$  do ▷ Loop over filter orders
     $\nabla h_q^{(l)} = \left( \mathcal{D}^{(1)}(h_q^{(l)}), \dots, \mathcal{D}^{(m)}(h_q^{(l)}) \right)^T$  ▷ Compute filters
     $\mathbf{h}^{(l)} \leftarrow \text{concat} \left( \mathbf{h}^{(l-1)}, \nabla h_1^{(l)}, \dots, \nabla h_m^{(l)} \right)$  ▷ Concatenate derivatives
end for
 $\mathbf{h}^{(l)} \leftarrow (\mathcal{E}_1(\mathbf{h}^{(l)}; \mathbf{A}_1), \dots, \mathcal{E}_c(\mathbf{h}^{(l)}; \mathbf{A}_c))$  ▷ Inner product features (optional)
 $\mathbf{z}_i \leftarrow \text{MLP}(\mathbf{h}^{(l)}; \omega)$  ▷ Pass through MLP

```

---

## S2 Distributional distance between latent representations

To test whether shifts in the statistical representation of the dynamical system can predict global phenomena in the dynamics we define a similarity metric between pairs of vector fields  $\mathbf{F}_1, \mathbf{F}_2$  with respect to their corresponding embeddings  $\mathbf{Z}_1 = (\mathbf{z}_{1,1}, \dots, \mathbf{z}_{n_1,1})$  and  $\mathbf{Z}_2 = (\mathbf{z}_{1,1}, \dots, \mathbf{z}_{n_2,1})$ . We use the optimal transport distance between the empirical distributions  $P_1 = \sum_i^{n_1} \delta(\mathbf{z}_{i,1})$ ,  $P_2 = \sum_i^{n_2} \delta(\mathbf{z}_{i,2})$

$$d(P_1, P_2) = \min_{\gamma} \sum_{uv} \gamma_{uv} \|\mathbf{z}_{u,1} - \mathbf{z}_{v,2}\|_2^2, \quad (\text{S18})$$

where  $\gamma$  is the transport plan, a joint probability distribution subject to marginality constraints that  $\sum_u \gamma_{uv} = P_2$ ,  $\sum_v \gamma_{uv} = P_1$  and  $\|\cdot\|_2$  is the Euclidean distance.

## S3 Examples

We give here more details on the examples of the main text.

### S3.1 Van der Pol oscillator on parabola example

We used the following equations to simulate the Van der Pol system:

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= \mu(1 - x^2)y - x,\end{aligned}\tag{S19}$$

parametrised by  $\mu$ . If  $\mu = 0$ , the system reduces to the harmonic oscillator, if  $\mu < 0$ , the system is unstable and if  $\mu > 0$ , the system is stable and converges to a limit cycle. In addition, we map this two-dimensional system to a parabola as with the map

$$\begin{aligned}x, y &\mapsto x, y, z = \text{parab}(x, y) \\ \dot{x}, \dot{y} &\mapsto \dot{x}, \dot{y}, \dot{z} = \text{parab}(x + \dot{x}, y + \dot{y}) - \text{parab}(x, y),\end{aligned}$$

where  $\text{parab}(x, y) = -(\alpha x)^2 - (\alpha y)^2$ . For Fig. 1, we trained MARBLE in the geometry-agnostic mode, see Table 1 for parameters. In Fig. S6, we illustrate other examples of the application of MARBLE on this system, with different regimes of  $\mu$ , and randomness in the parabola curvature across  $\mu$ , demonstrating the difference between geometry-aware and geometry-agnostic modes of MARBLE.

### S3.2 Low-rank RNNs

We consider low-rank RNNs composed of  $N = 500$  rate units in which the activation of the  $i$ -th unit is given by

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^N J_{ij} \phi(x_j) + \tilde{u}_i(t) + \eta_i(t), \quad x_i(0) = 0,\tag{S20}$$

where  $\tau = 100$  ms is a time constant,  $\phi(x_i) = \tanh(x_i)$  is the firing rate,  $J_{ij}$  is the rank-R connectivity matrix,  $u_i(t)$  is an input stimulus and  $\eta_i(t)$  is a white noise process with zero mean and  $\text{std } 3 \times 10^{-2}$ . The connectivity matrix can be expressed as

$$\mathbf{J} = \frac{1}{N} \sum_{r=1}^R \mathbf{m}_r \mathbf{n}_r^T,\tag{S21}$$

for vector pairs  $(\mathbf{m}_r, \mathbf{n}_r)$ . For the delayed-match-to-sample task, the input is of the form

$$\tilde{u}_i(t) = w_{1i}^{\text{in}} u_1(t) + w_{2i}^{\text{in}} u_2(t),\tag{S22}$$

where  $w_{1i}, w_{2i}$  are coefficients controlling the weight of inputs  $u_1, u_2$  into node  $i$ . Finally, the network firing rates are read out to the output as

$$o(t) = \sum_{i=0}^N w_i^{\text{out}} \phi(x_i).\tag{S23}$$

To train the networks we followed the procedure in<sup>50</sup>. The experiments consisted of 5 epochs; a fixation period of length between 100 – 500 ms chosen uniformly at random, a 500 ms stimulation period, a delay period of length between 500 – 3000 ms, a 500 ms stimulation period and a 1000 ms decision period. During training, the networks were subjected to the two inputs, whose value varied discontinuously between zero and a non-zero gain during stimulation periods. The networks were trained against a loss function

$$\mathcal{L} = |o(T) - \hat{o}(T)|,\tag{S24}$$

where  $T$  is the length of the trial and  $\hat{o}(T) = 1$  when both stimuli were present and  $-1$  otherwise. Coefficient vectors were initially drawn from zero-mean and unit std Gaussian distributions and then optimised. For training, we used the ADAM optimiser<sup>63</sup> with moment decay rates 0.9 – 0.999 and learning rates  $10^{-3} - 10^{-2}$ . We trained MARBLE in geometry-agnostic mode. See Table 1 for the parameters.



### S3.3 Macaque hand reaching data

We used publically available<sup>24</sup> electrophysiology data during a centre-out instructed-delay reaching task. The neural activity was recorded using linear multielectrode arrays (V-Probe, 24-channel linear probes) from rhesus macaque motor (M1) and dorsal premotor (PMd) cortices. See<sup>24</sup> for further details on experimental procedures. Each trial began with the monkey’s hand at the centre position, after which one (or more) of the radial targets at 10cm from the centre position were marked visually (target onset). After a variable delay period, one of the radial targets was highlighted indicating the go-cue for reaching. We analysed the 700ms period after the go-cue consisting of a delay followed by the reach. A total of 44 consecutive experimental sessions with a variable number of trials were considered.

For each of the 24 channels of each trial, we extracted the spike trains using the neo package in Python (<http://neuralensemble.org/neo/>) and converted them into instantaneous rates using Gaussian kernel convolution with a standard deviation of 100ms. We then binned the rates at 20ms intervals using the elephant Python package<sup>64</sup> to match the sampling frequency in the decoded kinematics in<sup>24</sup>. Finally, we fitted a PCA across all trials in a single session to reduce the dimension of the 24-channel rates to five principal components. We trained MARBLE in geometry-aware mode (without inner product features) separately on each individual session, treating each of the seven movement conditions as individual manifolds. See Table 1 for parameters.

To decode the hand kinematics from the neural trajectories, we followed the same procedure as in<sup>24</sup>. Specifically, we used Optimal Linear Estimation (OLE) to decode both the  $x$  and  $y$  reaching coordinates and velocities from the neural embeddings. For each individual session, using 5-fold cross-validation, we fitted an OLE to map from the MARBLE embeddings to the kinematics. To assess the accuracy of decoded movements, we computed the goodness of fit ( $R^2$ ) between the decoded and real velocities for both  $x$  and  $y$ , before taking the mean across them. We also trained a support vector machine classifier (regularisation of 1.0 with a radial basis function) on the real kinematic movements against the known condition labels. We then evaluated the classifier on the decoded kinematics and report the accuracy for each session.

Table 1: Parameters used for MARBLE embedding in the different experiments

Parameters	Experiment		
	van der Pol	Low-rank RNN	Macaque reaching
Inner product features	True	False/True	False
$k$	20	15	30
Diffusion	False	False	True
Feature order, $p$	2	2	2
MLP hidden channels	32	32	100
MLP output channels	5	3	3 or 20
Number of parameters	731	2886/1339	15803

## S4 Supplementary figures

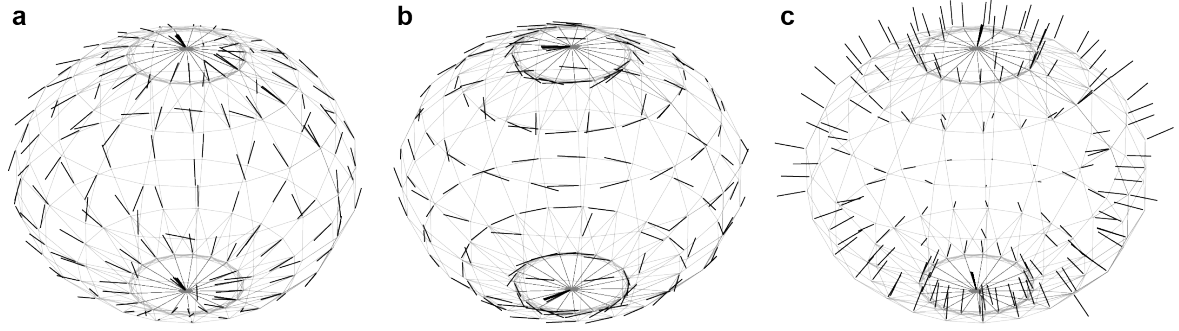


Figure S1: **Illustration of local coordinate frames.** **a,b** Local coordinate frames were fitted to eight neighbours at each point on the rectangular grid over a sphere (manifold of dimension two) embedded into  $\mathbb{R}^3$ . Unit vectors representing within-manifold basis. **c** Unit vector represents normal to the manifold. Note that the orientation of the normal vectors is not necessarily consistent.

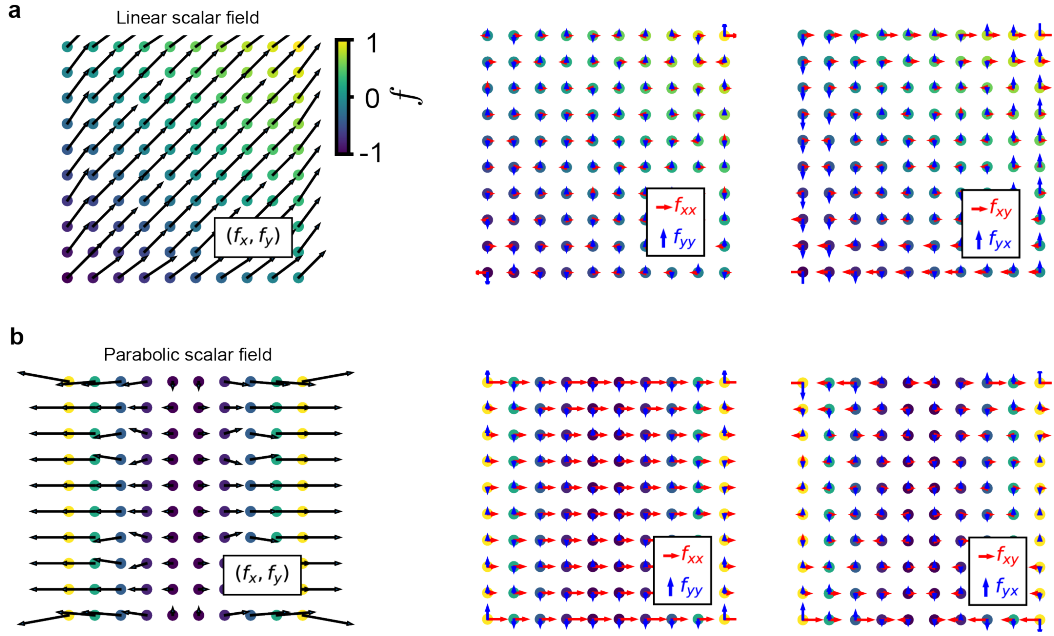


Figure S2: **Output of gradient filters.** **a** Scalar valued linear function. **b** Scalar valued parabolic function. The first column shows the output of the scalar signal convolved with the gradient filter with respect to a principal spatial coordinate, representing the directional (partial) derivatives along this direction. The second and third columns show second-order mixed partial derivatives obtained by a second application of the gradient filter to the derivative signal. In each case, we used a uniform rectangular grid with eight neighbours.

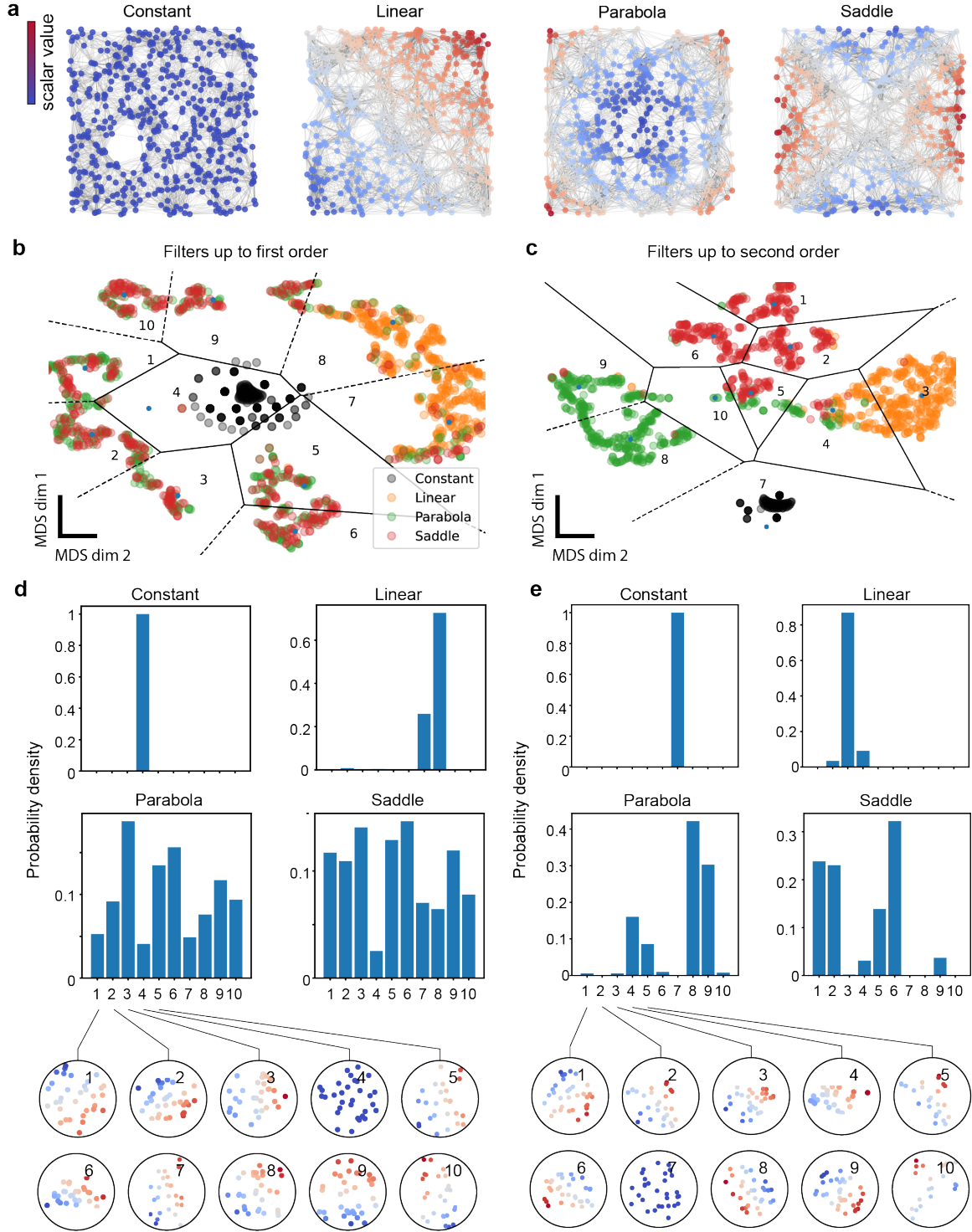


Figure S3: **Effect of filter order.** **a** Scalar functions sampled ( $n = 512$ ) uniformly at random and fitted with a continuous  $k$ -nearest neighbour graph ( $k = 15$ ). From left to right: constant, linear, parabola, saddle. **b** Joint embedding of local scalar fields from all functions based on first-order (1-hop) directional derivative filters. Dots represent nodes drawn from **a** with nodes close together signifying similar signal distributions in their neighbourhoods. Black lines show the convex polygon obtained from  $k$ -means clustering (10 clusters). **c** As in **b**, but with first (1-hop) and second-order (2-hop) filters. Including second-order filters increases the clustering of features. **d** Histogram representation of the clustered neighbourhood types. The latter is shown at the bottom in circular insets. **e** As in **d** but with first and second-order filters. Second-order features better discriminate the parabola and saddle but show little difference for constant and linear fields.

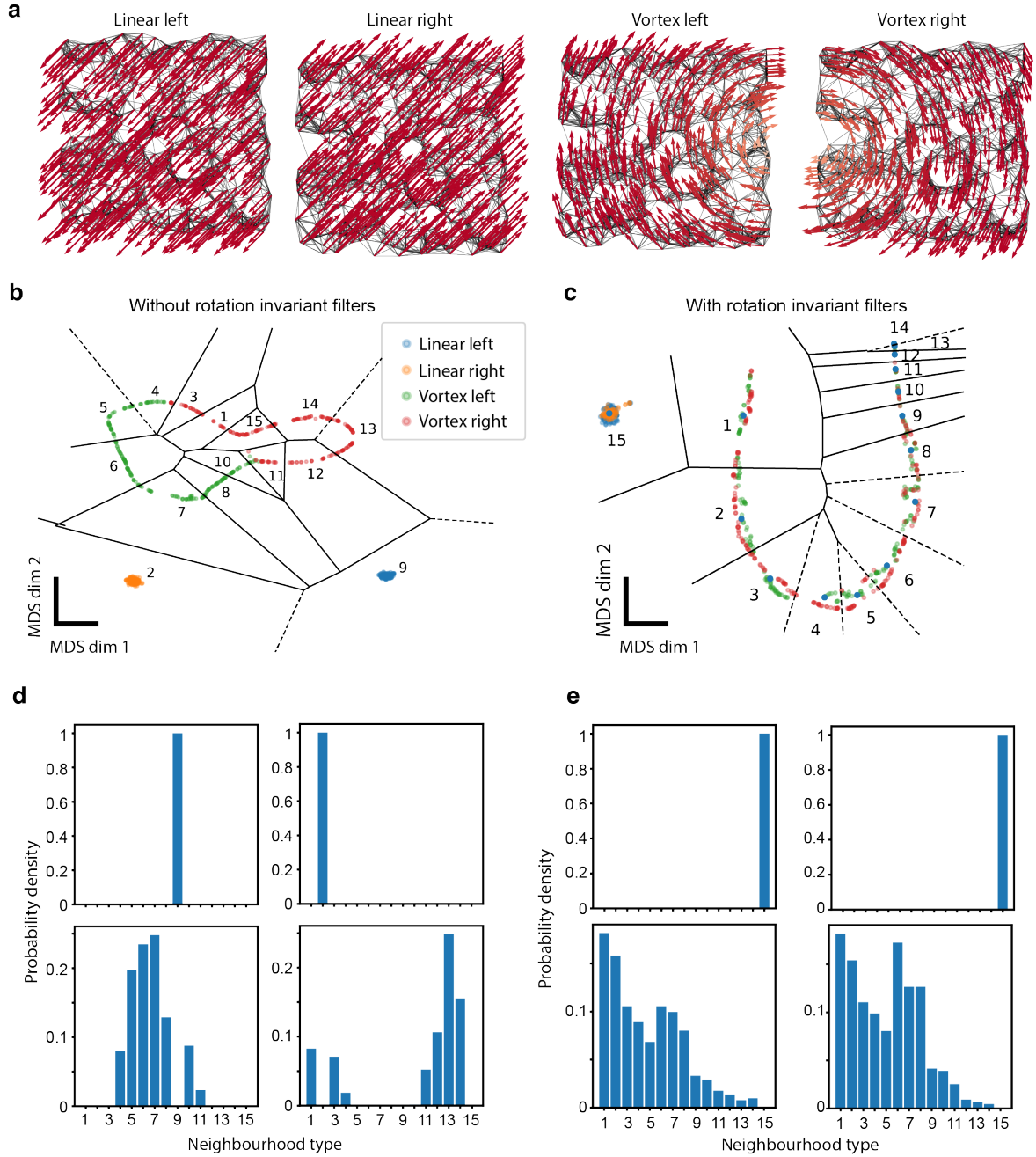


Figure S4: **Effect of rotational invariant filters.** **a** Vector fields sampled ( $n = 512$ ) uniformly at random in the interval  $[-1, 1]^2$  and fitted with a continuous  $k$ -nearest neighbour graph. **b** Joint embedding of local vector fields based on first-order (1-hop) directional derivative filters. Dots represent nodes drawn from a with nodes close together signifying similar signal distributions in their neighbourhoods. Features from the linear vector fields cluster together (clusters 2 and 9) while those drawn from the vortex fields fall on separate halves of a one-dimensional circular manifold corresponding to the one-parameter (angle) variation between them. **c** Same as **b** but with rotation invariant features. Features from linear fields can no longer be distinguished (cluster 15) because the filter does not learn the orientation. Features from vortex fields fall on a linear one-dimensional manifold parametrised by the distance from the centre. **d** The distribution of orientation-preserving derivative features can distinguish all fields. **e** The distribution of rotation-invariant features can discriminate linear fields from vortex fields but not the orientation.

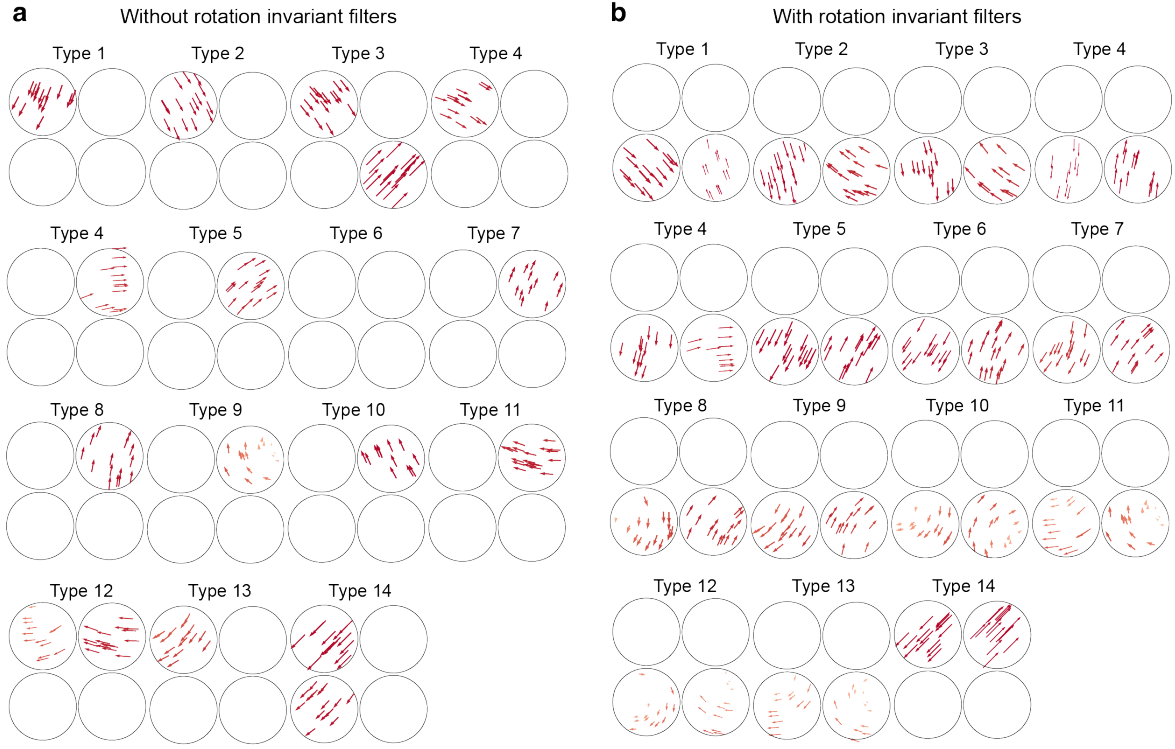


Figure S5: **Effect of rotational invariant filters on local vector field types.** Types are local vector field types with circular insets representing a local vector field drawn from each of the four fields in Fig S4. Top left: linear left; top right: linear right; bottom left vortex left; bottom right; vortex right. White insets indicate that the given type is not present in the vector field. **a** Without using rotation invariant filters the network tends to classify local vector fields into separate types. **b** With rotation invariant filters, local vector fields cluster more irrespective of the orientation. The separate types are instead distinguished more based on the expansion, and rotation properties of the features.

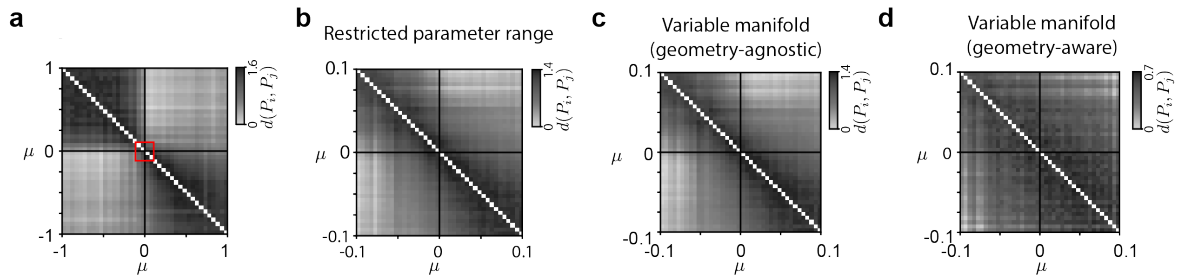
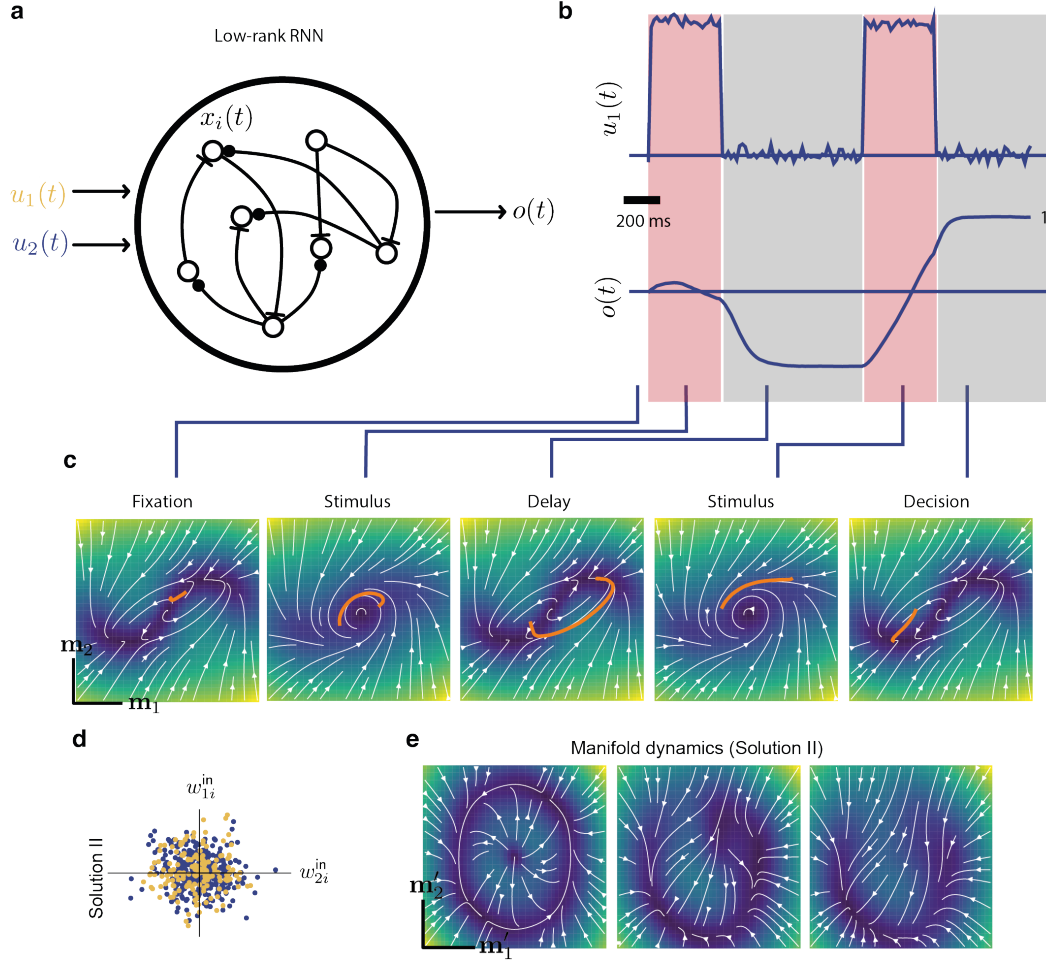
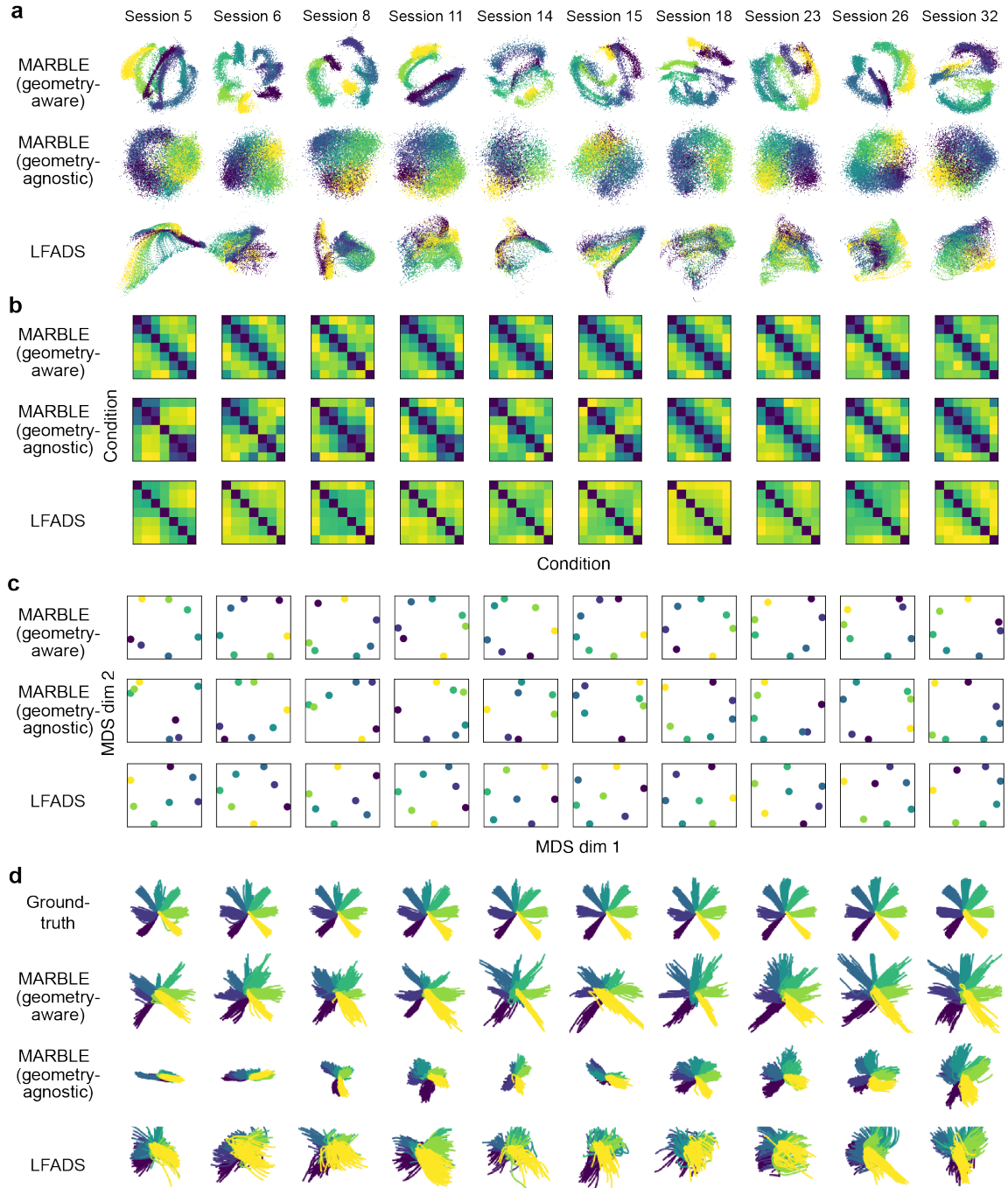


Figure S6: **Additional Van der Pol examples** We illustrate the MARBLE embedding of the Van der Pol example with 40 values of  $\mu$  in the following cases. **a** Using a larger parameter range  $\mu \in [-1, 1]$  increases the definition of the clusters. **b** Using a smaller parameter range  $\mu \in [-0.1, 0.1]$ , corresponding to the red square in a shows lower definition. **c** In geometry-agnostic mode, varying the curvature of the parabolic manifold  $\beta(x^2 + y^2)$  by drawing  $\beta$  uniformly at random from  $[-0.2, 0.2]$  does not alter the MARBLE embedding. **d** In geometry-aware mode, the same variation as in c destroys the cluster structure.





**Figure S7: Neural activities in a rank-2 RNN during the delayed-match-to-sample task.** **a** Schematic of a low-rank RNN, taking as input two stimuli and producing a decision variable as the output. Arrow endings represent inhibitory and excitatory connections. **b** Two example input patterns for one of the stimuli, and corresponding output patterns. Red and grey-shaded bands show stimulated and unstimulated periods. **c** Mean-field dynamics (heatmap and stream plot) superimposed with a sampled trajectory (orange) during one trial.



**Figure S8: Comparison of MARBLE and LFADS for learning neural dynamics for individual sessions of macaque reaching task.** **a** Feature embeddings obtained from MARBLE for various example sessions. The 3D-embedded points better reflect the arrangement of reaches in physical space when compared to LFADS. **b** The matrix of distribution distances between pairwise conditions for separate sessions shows a stronger periodic structure compared to LFADS. **c** MDS embedding of the distance matrix consistently recovers the spatial arrangement of reaches across sessions, when compare to LFADS. **d** Hand trajectories linearly decoded from MARBLE embeddings showed much stronger spatial correspondence to ground-truth kinematics than LFADS.