GENERATIVE MANIFOLD NETWORKS FOR EXPLAINABLE PREDICTION AND SIMULATION OF COMPLEX SYSTEM DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Manifold Networks (GMN) are a new machine learning framework consisting of a network of linked dynamical systems capturing causal interactions at the core of complex systems. The network is discovered by an interaction function which can focus on causality, shared information, nonlinearity or other discrimination metric. Network nodes are interactive low–dimensional data–driven state space generators accommodating multiscale dynamics. In contrast to many machine learning approaches GMN has no latent or random variables, operates solely on observed time series and thus provides explainability. GMN generates short and long term chaotic dynamics on par with echo state networks but at a remarkably reduced number of dimensions and without sensitive dependence on reservoir parameters or random states. As a result of the multiscale representation GMNs are able to learn the complete dynamics of a complex system based on limited training data. We demonstrate these features on chaotic dynamics and neural and behavioral recordings of the fruit fly *Drosophila melanogaster*.

1 INTRODUCTION

Network structure is a foundation of complex systems as demonstrated in genomic Turner et al. (2014), metabolic Jeong et al. (2001), physiologic Bashan et al. (2012), social, and neural systems/networks Bae et al. (2025); Assaf et al. (2020). Networks can enforce a remarkably low–dimensional structure within a high–dimensional system, consistent with the manifold hypothesis Thibeault et al. (2024) evidenced in living systems Eckmann & Tlusty (2021) and their neural processing Fontenele et al. (2024). The fact that fantastically complex structures such as mammalian brains express function and behavior not as single, ultra high–dimensional objects, but as interacting networks suggests the computational architecture should encompass low–dimensional, multiscale, interacting networks.

Generative manifold networks (GMN) combine these essential features into a new architecture based on interactive dynamical manifolds. GMN networks are discovered through an interaction function between observables defining an adjacency matrix from which a network graph for a desired target observable is grown. The interaction function can be a metric of causality such as convergent cross mapping (CCM) Sugihara et al. (2012), mutual information, nonlinearity Pao et al. (2021); Sugihara (1994); Smith (2015) or other suitable interaction metric. Each node of the network is a multivariate state space manifold leveraging the power of generalized embedding Deyle & Sugihara (2011). The architecture is therefore simple, low-dimensional and observable.

We demonstrate GMN applied to chaotic dynamics and *Drosophila* neural data with results compared to echo state networks (ESN). A brief review of reservoir computing (RC) and echo state networks with discussion of their connection to random embeddings and Koopman operators is provided in Appendix A.2.

1.1 GMN ARCHITECTURE

The GMN architecture is simple: a network of nodes, each node a dynamical system generator operating on a multivariate generalized embedding of observations where each node corresponds to a system observable. Here, we use the state space simplex generator defined by Sugihara & May (1990), however, other generators can be used. The generalized embedding contains multivariate inputs from other nodes while allowing time delayed versions of observed variables to represent hidden dynamics or unobserved variables Deyle & Sugihara (2011); Takens (1981). Time delay components are determined by node-specific embedding dimension E and time delay τ. Thus the network is inherently multiscale according to node embedding parameters, further, the manifold is entirely data–driven and observable, there are no latent variables as any delayed components representing an unobserved variable are direct mappings to observations. Additionally, statistical model constraints such as independence and linearity are not imposed. A schematic of GMN with comparison to a reservoir computer is shown in figure 1.

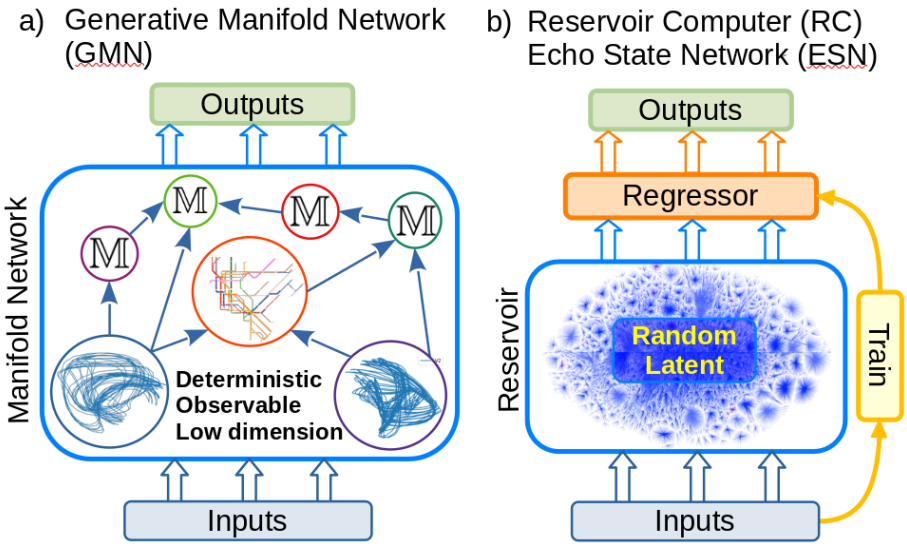

Figure 1: Schematic representation of a) generative manifold networks (GMN) and b) reservoir computer (RC). RC is predicated on an extremely high–dimensional random embedding (reservoir) with sensitive dependence on reservoir size, distribution and parameters. GMN encapsulates interactive dynamical manifolds, represented in the graphic by an embedding, map or symbol $\mathbb{M}$, each a deterministic, observable, low–dimensional system in $\mathbb{R}^E$.

2 RESULTS

2.1 CHAOTIC DYNAMICS: LORENZ '63

The Lorenz'63 atmospheric convection model defines a 3–dimensional manifold according to $\frac{dx}{dt} = a(y-x)$, $\frac{dy}{dt} = x(b-z) - y$, $\frac{dz}{dt} = xy - cz$ and with parameters a=10, b=28, c=8/3 generates chaotic dynamics. We generate a test set of V1 = x, V2 = y, V3 = z of 4000 points starting at time 10.0 with $\Delta t = 0.02$.

We define a simple 3 node GMN for V1, V2, V3 where node V1 feeds output to V2 and V3, node V2 feeds output to node V3 as shown in figure 3a. We also create an echo state network with 3 inputs and

3 outputs for V1, V2, V3, and 1000 reservoir nodes. Implementation details are described in Methods. Both generators are trained on the first 2000 points of data with free-running generation starting at Time = 50 (index 2001) with results shown in figure 2 where the 1000 node RC achieves good accuracy for approximately 2 seconds. Small improvements can be made with a larger reservoir ESN, but 1000 nodes illustrates reasonable synchronization with the true dynamics. The GMN achieves a comparable but longer period of approximately 2¾ seconds demonstrating GMN is capable of short term accuracy in the generation of chaotic dynamics.

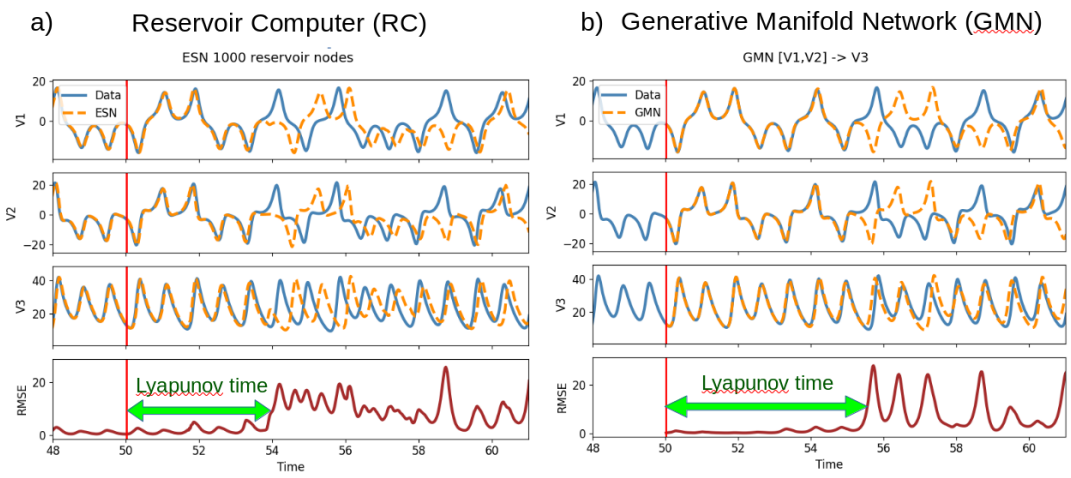

Figure 2: Generated dynamics of the Lorenz'63 3D chaotic dynamical system by a 1000 node ESN and 3 node GMN. Generative mode begins at Time = 50.

Next we examine long term generation by training both GMN and ESN on the first 2000 points then generating 1000 points with results presented in figure 3 where the 3 node GMN generates dynamical behavior consistent with the known dynamics while the ESN requires 3000 reservoir nodes to achieve similar results. To compare the properties of the generated time series we calculated the power spectrum of the generated time series indicating the ESN creates unnatural high frequency components. The power of these components decrease with reservoir size but still depart significantly from the spectrum of GMN which is closest to the original Lorenz dynamics (Fig. 3e).

Figure 4 demonstrates the importance of proper manifold interaction for the Lorenz'63 system. In figure 4a the 3 node network has only two connections supplying interactions from V1 to V3 and V2 to V3, no interaction between V1 and V2. Adding an interaction from V1 to V2 enables full recovery of the system dynamics.

2.2 *Drosophila*

Next we compare GMN and ESN on neural recordings of a fly expressing the calcium indicator GCaMP6f as a measure of neuronal activity. The fly was imaged walking on a freely rotating Styrofoam ball allowing recording of forward speed (FWD) and left/right turning speed (Left_Right) as shown in figure 5. Neural activity was spatially segmented by independent component analysis (ICA) yielding 80 component brain areas and corresponding time series for a total of 82 observables. Details are provided in Aimon et al. (2019).

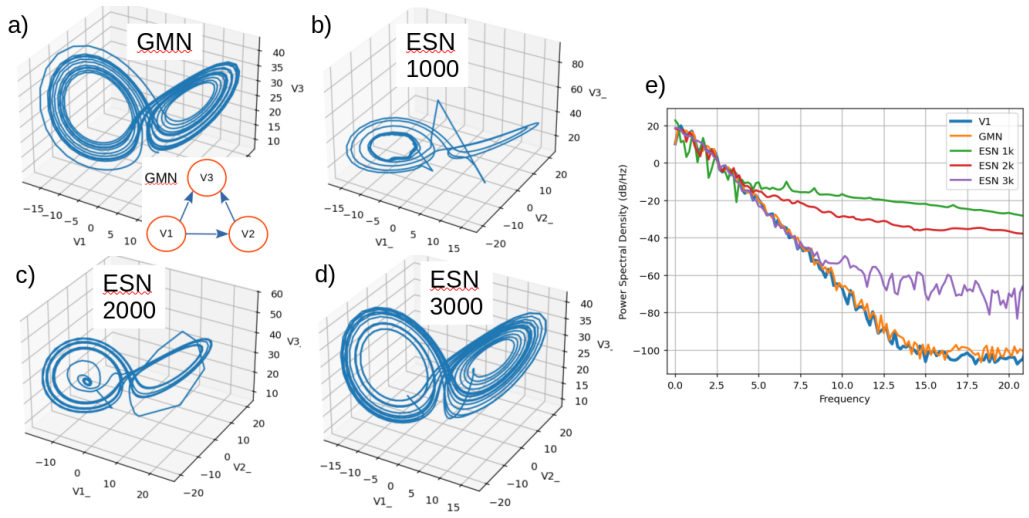

Figure 3: Long term generated dynamics of the Lorenz'63 3D chaotic dynamical system by a) 3 node GMN, b) 1000 node, c) 2000 node, and d) 3000 node ESN. e) Power spectral density of variable V1 and generated versions indicating the examined ESN generate high–frequency components not present in the data.

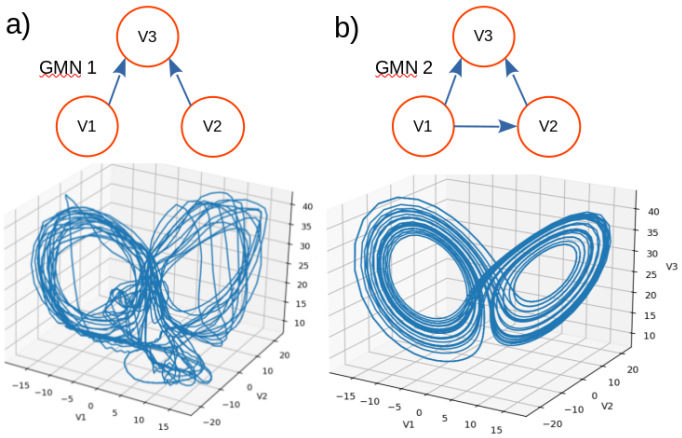

Figure 4: a) An incomplete GMN network of the Lorenz'63 system fails to generate valid dynamics. b) A complete network generates full dynamics.

The GMN network is created from an interaction matrix based on a measure of nonlinearity $\rho_\Delta(x,y) = \rho_{CM}(x,y) - |\rho_P(x,y)|$ where $\rho_{CM}(x,y)$ is simplex nonlinear cross map correlation and $\rho_P(x,y)$ the linear Pearson correlation between observables x,y. Starting from the forward motion observable (FWD) the network is grown according to the interaction/adjacency matrix as described above and in Methods resulting in a 67 node network shown in figure 5c.

Comparison of GMN and ESN results are depicted in figure 6 where training data span index 1 - 6000, generative dynamics start at index 6001. GMN is run with $E = 7$, $\tau = -8$ for all nodes. The training set

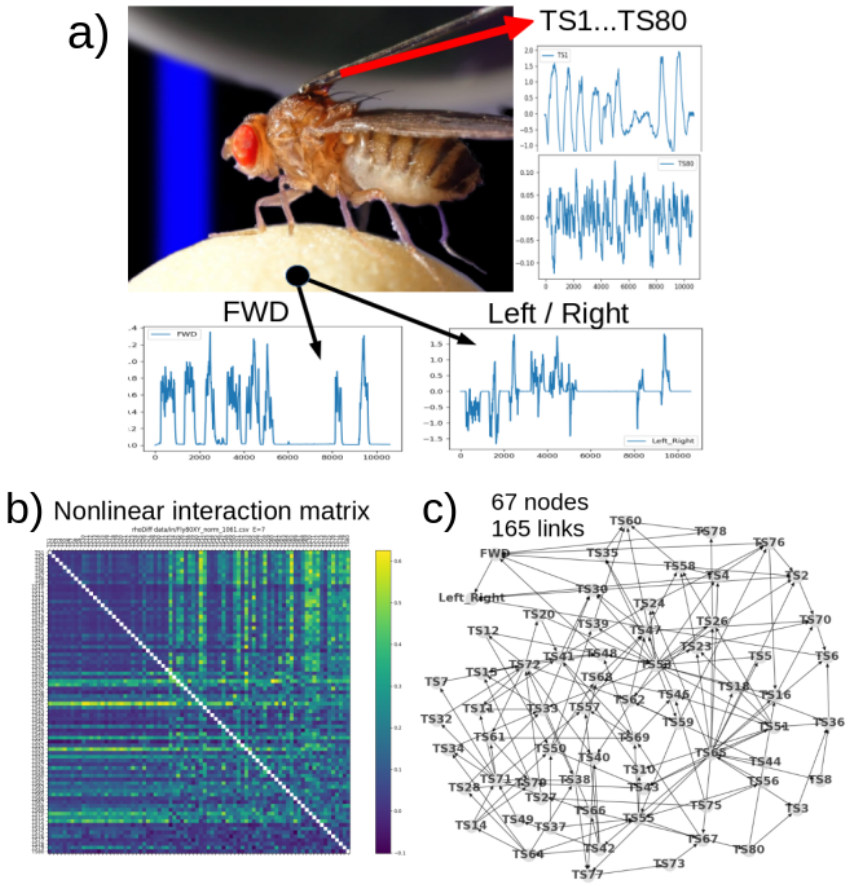

Figure 5: a) Neural and behavioral measurements of *Drosophila* walking on a Styrofoam ball. b) Nonlinear interaction matrix of all time series used to create the GMN network. c) GMN network.

consists of six bouts of forward motion separated by small time intervals and does not include the protracted pause in forward motion between the sixth and seventh forward movements. Remarkably, GMN generated dynamics reproduce this pause behavior demonstrating GMN can produce realistic behaviors that are not explicitly present in the training set. This is possible from interactions of multivariate, multiscale generalized embeddings that capture system dynamics across scales with multiple variables as well as delays thereof. Removal of either of these features, node interactions or multiscale embedding, precludes this ability as verified with autonomous generation of dynamics from a univariate time delay embedding (no network interactions) and from multivariate embeddings with $E = 1$ (no multiscale dynamics) that do not generate the observed pause. We therefore infer GMN are capable of learning the complete dynamics of a system as long as the observations are sufficiently informative, the network structure sufficiently connected, and time delays appropriate.

In contrast, ESN generation of *Drosophila* forward motion fails to reproduce characteristics of the observed time series in the withheld data. The ESN generated time series start immediately with a fast bout then producing multiple bouts in a quick succession rather than a long pause and two bouts in the withheld data.

A direct comparison of GMN & ESN generation is shown in figure 6d where the ESN is observed to generate unrealistic negative forward motions while GMN generated dynamics better approximate the withheld data.

It should be noted that in principle the ESN result can be improved, perhaps with larger reservoirs, deep ESN, or next generation ESN (adding explicit nonlinear outputs or internal states) Zhang & Vargas (2023), however, one is then engaged in a protracted optimization exercise with a reservoir that may be finely tuned to the specific behavior Zhang & Cornelius (2023), and, one has still not recovered mechanistic insight derived from observables. As noted earlier. in contrast to most machine learning approaches GMN is based solely on observables with no latent or random variables facilitating transparency, explainability and hypothesis testing.

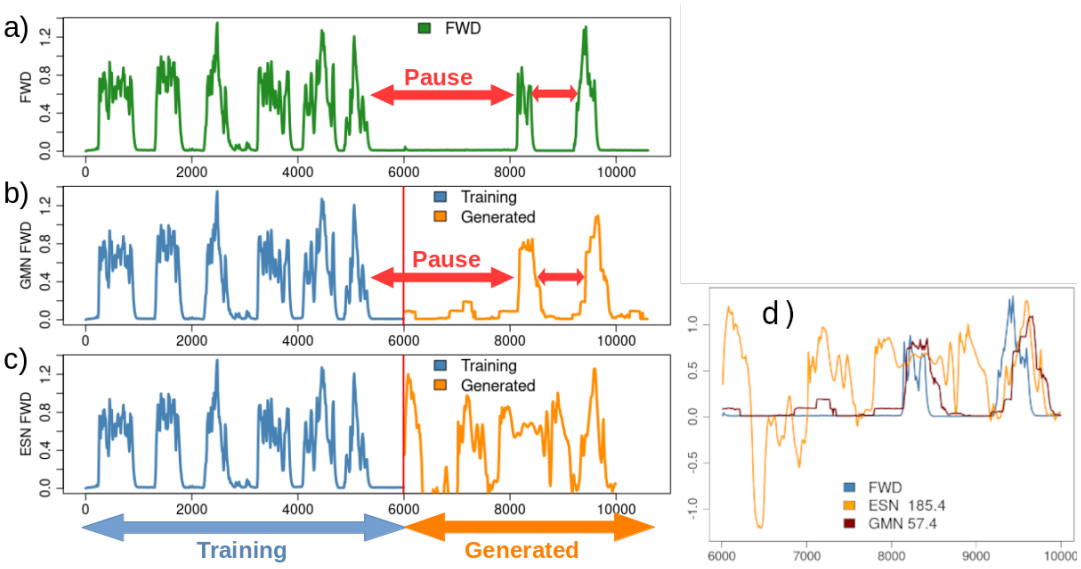

Figure 6: Comparison of *D*rosophila FWD motion generated by a 3000 node ESN and the 67 node GMN. Generative mode begins at index 6001. a) Observed forward motion (FWD). b) GMN generated FWD dynamics. c) ESN generated FWD dynamics. d) Overlay comparison. The fly paused for an extended period before proceeding with two bouts of forward movement. ESN immediately generates multiple rapid high speed bouts and negative FWD motion not present in the real data, whereas the GMN generated time series captures the pause behavior and the two bouts of forward movement. Cumulative absolute error (CAE) of ESN with respect to FWD is 185.4, GMN 57.4.

3 METHODS

Generative manifold networks are conceptually and operationally simple consisting of a network of interacting dynamical systems generating time series at each node of the network. GMN is data–driven with the network discovered from an interaction/adjacency matrix, and node generation by a multi-input / single-output (MISO) function. GMN can therefore be completely configured with the choice of interaction and generator functions.

The network interaction function can be any comparative or discrimination function such as convergent cross mapping, nonlinearity or correlation. Several candidate interaction functions are detailed in Appendix A.1. In the *Drosophila* example we use a nonlinearity metric $\rho_{\triangle}(x, y)$ detailed in section 2.2.

Here, nodes consist of generalized embeddings Deyle & Sugihara (2011) with a simplex state space generator Sugihara & May (1990) characterized with two parameters, the embedding dimension E and time delay τ. However, each node can have a distinct generator with node-specific parameters. Values of E and τ can be determined by assessing optimal simplex predictive skill of the target variable over ranges of E and τ.

Results presented here are computed with the python GMN package Biological Nonlinear Dynamics Data Science Unit, OIST (2023) and EDM package Sugihara Lab, Scripps Institution of Oceanography (2025). Code and data to reproduce the results are available at GMN_ESN_Examples.

3.1 Algorithm Overview

Given an N row by M column observation matrix $\mathrm{O_{NM}}$ where M corresponds to observables and N time series observations, an MxM interaction matrix $\mathrm{I_{MM}}$ is computed. The associated GMN is implemented in two steps:

1. Create Network
   - ☐ Starting at a desired target node selected from the M observables, add up to $\mathrm{N_D}$ driver nodes for the target according to the interaction/adjacency matrix while disallowing network cycles.
   - ☐ Repeat at each connected node until no more connections.
2. Generate Dynamics
   - ☐ The network state space manifold $\mathbb{M}$ is created from a specified time range of observations.
   - ☐ Based on the manifold, a forward prediction is made at each node. The set of predictions from all nodes constitutes the network output at that time step, the one-step ahead manifold $\widehat{\mathbb{M}}$.
   - ☐ Forward predictions are repeated for a defined number of steps. The previous time step network output defines the state from which the next time step prediction will be made.

3.2 Algorithms

3.2.1 Create Network

Given an interaction matrix $\mathrm{I_{MM}}$, target observation node, node functions and number of node drivers (inputs) create the network graph G, see algorithm 1.

Algorithm 1 Create Network

Require: $\mathrm{I_{MM}}, \mathrm{F_N}(), \mathrm{N_D}$ ▷ *Interaction matrix, Node function(s), N drivers*
nodes ← $\mathrm{I_{MM}}$.columns ▷ *List of nodes, target first*
G ← DiGraph() ▷ *Instantiate directed graph*
while nodes **do**
  node ← nodes.pop ▷ *Remove node from nodes*
  topDrivers ← $\mathrm{I_{MM}}$[node].sort[: $\mathrm{N_D}$] ▷ *top* $\mathrm{N_D}$ *drivers for node*
  addedDrivers ← [] ▷ *Empty list*
  for driver in topDrivers **do**
    if G.acyclic(node) **then**
      G.add_node(node) ▷ *node does not add cycle, add to G*
      addedDrivers.append(node)
  for driver in addedDrivers **do**
    nodes.append(driver) ▷ *Add driver for upstream nodes*
output G

3.2.2 GENERATE

Given a GMN graph G, observation matrix $\mathrm{O_{NM}}$ and training indices, generate dynamics, see algorithm 2.

Algorithm 2 Generate

Require: $\mathrm{G}, \mathrm{O_{NM}}, \mathrm{train}, N$ ▷ *GMN, observation matrix, training indices, N generated*
$\mathbb{M} \leftarrow \mathbb{E}(\mathrm{O_{NM}}, \mathrm{train})$ ▷ *Manifold library from observations*
$\widehat{\mathbb{M}} \leftarrow \varnothing$ ▷ *Generated Manifold, init empty*
for t in 1:N **do** ▷ *For all time steps*
  nodeOut ← [] ▷ *Array for node outputs*
  for node in G.nodes.TopologicalSorted **do**
    nodeOut[node] ← G[node].Generate($\mathbb{M}$,$\widehat{\mathbb{M}}$) ▷ *Generate*
  $\widehat{\mathbb{M}[t+1]}$ ← nodeOut ▷ *Next time step state for next generation*
output $\widehat{\mathbb{M}}$

4 DISCUSSION

Complex systems often express a radical reduction of free parameters/dimensions in functional networks/manifolds. Generative manifold networks capture this through a network of interacting manifolds that assess information flow to a target observable under conditions of interest such as causality, mutual information or nonlinearity, while each manifold accommodates multivariate, multiscale dynamics through generalized embedding. Manifold generators can be any suitable multivariate function capturing multiscale dynamics. If the generators are data–driven the architecture provides an observable, explainable, testable representation without latent or random variables. It should also be noted that at each generated time step not only are values produced for the target variable, but for each node in the network providing holistic generation of all observables.

To corroborate multiscale dynamics in GMN we note that GMN without multiscale temporal information fails to reproduce the complete dynamics of forward motion observed in fly behavior with a long pause. In this case inclusion of time delays is essential implying that critical information is either not observed in the recordings or that additional timescale should be included to capture properties of the system. We also infer network structure and manifold interactions are required for complete dynamical representation, a single manifold fails to exhibit the observed behavioral pause.

4.1 GENERALIZED APPLICATION & SCALABILITY

The GMN architecture is data–driven and should be applicable to any multivariate observed system. In addition to the work reported here GMN has been applied to electrophysical neural recording of rats, neural dynamics from whole brain calcium imaging of larval zebrafish, fMRI of human brain activity experiencing a virtual reality simulation, and Antarctic sea ice dynamics. Since GMN generates values for all observables in the network it can function as a comprehensive simulator of every variable/observable of the system.

Regarding scalability we note GMN computational load scales linearly with the number of nodes and node dimension, having been successfully applied to neural data with 127,556 variables. Additionally, GMN has the option to use the Kokkos based kEDM backend for numerical computations enabling GPU accelerated computations.

4.2 ARE GMN UNIVERSAL?

The Johnson-Lindenstrauss lemma ensures that random embeddings of sufficiently high dimension can linearize any function, the basis of Koopmanism and Reservoir Computing Budišić et al. (2012). Indeed, echo state networks can be universal approximators and Koopman operator approximators Grigoryeva & Ortega (2018); Gulina & Mauroy (2021). Since a generalized embedding as used in GMN can be an arbitrarily high–dimensional random embedding, with sufficient dimension or network size GMN can by induction represent arbitrarily complex systems.

With demonstration that GMN operate on par with ESN in generation of short term synchronized chaotic dynamics, and can completely represent complex dynamics, we hypothesize that GMNs might also be universal approximators with a topological constraint. If borne out, this could be significant as GMN are entirely observable and explainable.

With repsect to the manifold hypothesis we are motivated to seek low–dimensional network and manifold expressions: Can one ensure a low dimensional network to represent a system of arbitrary complexity can be found? This question is currently open but empirical evidence of the manifold hypothesis as discussed in the introduction indicates it can.

4.3 FUTURE AVENUES

The current GMN implementation generates manifold values based on a static manifold $\mathbb{M}$ of a training set. Generated values at each time step are fed back as the starting state for the next generated value, however the prediction is found within the static manifold $\mathbb{M}$. We expect GMN can be extended to allow online learning where the manifold $\mathbb{M}$ is updated with new states or rules for dynamic, continual learning.

Complex system regulation via cybernetics is a central feature of persistent and homeostatic systems. Work is underway to adapt GMN with external feedback control to realize generalized model process control for complex systems.

Leveraging the significant dimensional reduction and transparent, explainable basis of GMN, it should find broad applicability as a surrogate model in coupled/hybrid model applications where surrogate (embedded) models of complex dynamics replace computationally intensive numerical model components Pestourie et al. (2023).

REFERENCES

Sophie Aimon, Takeo Katsuki, Tongqiu Jia, Logan Grosenick, Michael Broxton, Karl Deisseroth, Terrence J. Sejnowski, and Ralph J. Greenspan. Fast near-whole–brain imaging in adult drosophila during responses to stimuli and behavior. *PLOS Biology*, 17(2):1–31, 02 2019. doi: 10.1371/journal.pbio.2006732. URL https://doi.org/10.1371/journal.pbio.2006732.

Yaniv Assaf, Arieli Bouznach, Omri Zomet, Assaf Marom, and Yossi Yovel. Conservation of brain connectivity and wiring across the mammalian class. *Nature Neuroscience*, 23(7):805–808, 2020. doi: 10.1038/s41593-020-0641-7. URL https://doi.org/10.1038/s41593-020-0641-7.

J. Alexander Bae, Mahaly Baptiste, and Maya R. Baptiste et al. Functional connectomics spanning multiple areas of mouse visual cortex. *Nature*, 640(8058):435, 2025. doi: 10.1038/s41586-025-08790-w. URL https://doi.org/10.1038/s41586-025-08790-w.

Amir Bashan, Ronny Bartsch, Jan Kantelhardt, Shlomo Havlin, and Plamen Ch. Ivanov. Network physiology reveals relations between network topology and physiological function. *Nature Communications*, 3(1): 702, 2012. doi: 10.1038/ncomms1705. URL https://doi.org/10.1038/ncomms1705.

Biological Nonlinear Dynamics Data Science Unit, OIST. Generative manifold networks (gmn). Python Package Index, 2023. URL https://pypi.org/project/gmn/. Computer Software.

M. Budišić, R. Mohr, and I. Mezić. Applied koopmanism. *Chaos*, 22(4):047510, 2012. doi: 10.1063/1.4772195. URL https://doi.org/10.1063/1.4772195.

E. R. Deyle and G. Sugihara. Generalized theorems for nonlinear state space reconstruction. *PLoS One*, 6 (3):e18295, 2011. doi: 10.1371/journal.pone.0018295. URL https://www.ncbi.nlm.nih.gov/pubmed/21483839.

Xing-Yue Duan, Xiong Ying, Si-Yang Leng, Jürgen Kurths, Wei Lin, and Huan-Fei Ma. Embedding theory of reservoir computing and reducing reservoir network using time delays. *Phys. Rev. Res.*, 5:L022041, May 2023. doi: 10.1103/PhysRevResearch.5.L022041. URL https://link.aps.org/doi/10.1103/PhysRevResearch.5.L022041.

Jean-Pierre Eckmann and Tsvi Tlusty. Dimensional reduction in complex living systems: Where, why, and how. *BioEssays*, 43(9):2100062, 2021. doi: https://doi.org/10.1002/bies.202100062. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bies.202100062.

Antonio J. Fontenele, J. Samuel Sooter, V. Kindler Norman, Shree Hari Gautam, and Woodrow L. Shew. Low-dimensional criticality embedded in high-dimensional awake brain dynamics. *Science Advances*, 10 (17):eadj9303, 2024. doi: 10.1126/sciadv.adj9303. URL https://www.science.org/doi/abs/10.1126/sciadv.adj9303.

Lyudmila Grigoryeva and Juan-Pablo Ortega. Echo state networks are universal. *Neural Networks*, 108:495–508, 2018. doi: 10.1016/j.neunet.2018.08.025. URL https://doi.org/10.1016/j.neunet.2018.08.025.

Marvyn Gulina and Alexandre Mauroy. Two methods to approximate the koopman operator with a reservoir computer. *Chaos*, 31(2):023116, 2021. doi: /10.1063/5.0026380. URL https://doi.org/10.1063/5.0026380.

Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004. doi: 10.1126/science.1091277. URL https://www.science.org/doi/10.1126/science.1091277.

H. Jeong, S. Mason, and AL. Barabasi et al. Lethality and centrality in protein networks. *Nature*, 411(6833): 41–42, 2001. doi: 10.1038/35075138. URL https://doi.org/10.1038/35075138.

Zhixin Lu, Brian R. Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos*, 28(6): 061104–1, 2018. doi: 10.1063/1.5039508. URL https://doi.org/10.1063/1.5039508.

Akane Ohkubo and Masanobu Inubushi. Reservoir computing with generalized readout based on generalized synchronization. *Scientific Reports*, 14(1):30918, 2024. doi: 10.1038/s41598-024-81880-3. URL https://doi.org/10.1038/s41598-024-81880-3.

Gerald M Pao, Cameron Smith, Joseph Park, Keichi Takahashi, Wassapon Watanakeesuntorn, Hiroaki Natsukawa, Sreekanth H Chalasani, Tom Lorimer, Ryousei Takano, Nuttida Rungratsameetaweemana, and George Sugihara. Experimentally testable whole brain manifolds that recapitulate behavior, 2021. URL https://arxiv.org/abs/2106.10627.

Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos*, 27(12):121102–1, 2017. doi: 10.1063/1.5010300. URL https://pubs.aip.org/aip/cha/article/27/12/121102/135382/Using-machine-learning-to-replicate-chaotic.

Raphaël Pestourie, Youssef Mroueh, Chris Rackauckas, Payel Das, and Steven G. Johnson. Physics-enhanced deep surrogates for partial differential equations. *Nature Machine Intelligence*, 5(12): 1458–1465, 2023. doi: 10.1038/s42256-023-00761-y. URL https://doi.org/10.1038/s42256-023-00761-y.

Reginald Smith. A mutual information approach to calculating nonlinearity. *Stat*, 4(1):291–303, 2015. doi: https://doi.org/10.1002/sta4.96. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sta4.96.

G. Sugihara. Nonlinear forecasting for the classification of natural time series. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 348:477–495, 1994. doi: 10.1098/rsta.1994.0106. URL https://royalsocietypublishing.org/doi/10.1098/rsta.1994.0106.

G. Sugihara and R. May. Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series. *Nature*, 344:734–741, 1990. doi: 10.1038/344734a0. URL https://www.nature.com/articles/344734a0.

G. Sugihara, R. May, and H. Ye. Detecting causality in complex ecosystems. *Science*, 338:496–500, 2012. doi: 10.1126/science.1227079. URL https://www.science.org/doi/10.1126/science.1227079.

Sugihara Lab, Scripps Institution of Oceanography. Empirical dynamic modeling (edm). Python Package Index, 2025. URL https://pypi.org/project/pyEDM/. Computer Software.

F. Takens. Detecting strange attractors in turbulence. In D. A. Rand and L. S. Young (eds.), *Dynamical Systems and Turbulence, Lecture Notes in Mathematics*, volume 8, pp. 366–381. Springer-Verlag, 1981.

Vincent Thibeault, Antoine Allard, and Patrick Desrosiers. The low-rank hypothesis of complex systems. *Nature Physics*, 20(2):294–302, 2024. doi: 10.1038/s41567-023-02303-0. URL https://doi.org/10.1038/s41567-023-02303-0.

LM Turner, MA White, D Tautz, and BA Payseur. Genomic networks of hybrid sterility. *PLoS Genet*, 10(2):e1004162, 2014. doi: 10.1371/journal.pgen.1004162. URL https://doi.org/10.1371/journal.pgen.1004162.

Min Yan, Can Huang, Peter Bienstman, Peter Tino, Wei Lin, and Jie Sun. Emerging opportunities and challenges for the future of reservoir computing. *Nature Communications*, 15(1):2056, 2024. doi: 10.1038/s41467-024-45187-1. URL https://doi.org/10.1038/s41467-024-45187-1.

Heng Zhang and Danilo Vasconcellos Vargas. A survey on reservoir computing and its interdisciplinary applications beyond traditional machine learning. *IEEE Access*, 11:81033–81070, 2023. doi: 10.1109/ACCESS.2023.3299296. URL https://ieeexplore.ieee.org/document/10196105.

Yuanzhao Zhang and Sean P. Cornelius. Catch-22s of reservoir computing. *Phys. Rev. Res.*, 5:033213, Sep 2023. doi: 10.1103/PhysRevResearch.5.033213. URL https://link.aps.org/doi/10.1103/PhysRevResearch.5.033213.

A Appendix

A.1 GMN Interaction and Node functions

Table 1 lists candidate interaction functions to discover the manifold network.

Table 1: GMN Network interaction functions

Purpose	Functions
Nonlinearity	ρ_{Δ}Pao et al. (2021), $\mathrm{MI}_{\mathrm{NL}}$Smith (2015), S-map Sugihara (1994)
Shared information	Simplex cross map Sugihara & May (1990), Mutual information (MI)
Shared dynamics	Convergent cross mapping (CCM) Sugihara et al. (2012)
Separable dynamics	CCM with minimum MI
Separability	Clustering, PCA
Linear dependence	Correlation

Table 2 lists candidate generator functions currently available in the `gmn` package.

Table 2: GMN Node generator functions

Function	Reference
Simplex	pyEDM Simplex
S-map	pyEDM SMap
k-nearest neighbors	sklearn neighbors.KNeighborsRegressor
Support Vector Regression	sklearn.svm SVR
Linear	sklearn.linear_model LinearRegression

A.2 Reservoir computers

Reservoir computing (RC) is a vibrant field with recent reviews provided by Zhang & Vargas (2023); Yan et al. (2024). Recognition of the echo state property Jaeger & Haas (2004) gave rise to echo state networks (ESN) which were celebrated for their ability to accurately generate short term chaotic dynamics Pathak et al. (2017); Lu et al. (2018) as well as generating long term dynamics consistent with the underlying manifold. The reservoir represents a high–dimensional random embedding and as such there is an affinity to Koopman operators. A central tenant of Koopmanism is that within an infinite dimensional space a linear

approximation to any nonlinear dynamic may be found, and RC have been shown to be Koopman operator approximators Gulina & Mauroy (2021).

However, as demonstrated by Pathak et al. (2017) the output is highly sensitive to the reservoir structure and parameters. Thus one trades ease of construction without reservoir training with optimization of reservoir structure and parameters. Nonetheless, the ease with which an ESN can be configured and trained (conventionally, only the output layer is trained) coupled with recognition that they are universal approximators Grigoryeva & Ortega (2018) has fueled their application.

Recent developments focus on expanding the capability and information content of the trainable output layer to better represent nonlinear dynamics Zhang & Cornelius (2023); Ohkubo & Inubushi (2024) or reduce reservoir dimensionality Duan et al. (2023). While these approaches are demonstrated to improve RC performance under specific conditions, conceptually one can question why a universal approximator needs nonlinear output modulation or storage of dynamical information in the output layer. This is likely to introduce fragility and it is perhaps better to represent dynamics in the reservoir. Indeed, it is recognized that RC require substantial warmup (training) to capture attractor dynamics, and even though next generation reservoir computers (NGRC) address this by adding output nonlinearites, they are critically sensitive to the choice of readout nonlinearity Zhang & Cornelius (2023).