Do We Need an Encoder-Decoder to Model Dynamical Systems on Networks?

Bing Liu[1,2] , **Wei Luo**[3] , **Gang Li**[3] , **Jing Huang**[1,2,*] and **Bo Yang**[1,2*]

[1]College of Computer Science and Technology, Jilin University, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China
[3]School of Information Technology, Deakin University, Geelong, Australia
liubing20@mails.jlu.edu.cn, {wei.luo, gang.li}@deakin.edu.au, {huangjing, ybo}@jlu.edu.cn

Abstract

As deep learning gains popularity in modelling dynamical systems, we expose an underappreciated misunderstanding relevant to modelling dynamics on networks. Strongly influenced by graph neural networks, latent vertex embeddings are naturally adopted in many neural dynamical network models. However, we show that embeddings tend to induce a model that fits observations well but simultaneously has incorrect dynamical behaviours. Recognising that previous studies narrowly focus on short-term predictions during the transient phase of a flow, we propose three tests for correct long-term behaviour, and illustrate how an embedding-based dynamical model fails these tests, and analyse the causes, particularly through the lens of topological conjugacy. In doing so, we show that the difficulties can be avoided by not using embedding. We propose a simple embedding-free alternative based on parametrising two additive vector-field components. Through extensive experiments, we verify that the proposed model can reliably recover a broad class of dynamics on different network topologies from time series data.

1 Introduction

In deep learning, architectural design patterns proven successful for one task are often transplanted to tackle related tasks. One such design pattern is an *encoder-decoder* structure found in almost every competitive model, be it a transformer [Vaswani *et al.*, 2017] or a graph neural network [Kipf and Welling, 2016]. With recent dramatic progress in neural dynamical model [Chen *et al.*, 2018; Chamberlain *et al.*, 2021], one would naturally assume that an encoder-decoder structure in such models is also helpful and necessary.

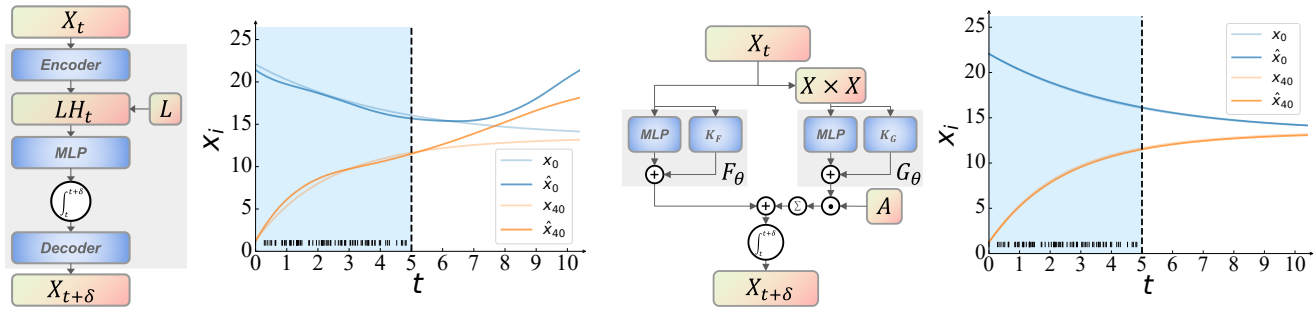
We challenge this assumption in the context of reconstructing network dynamics from observation time series. Many natural and social phenomena can be modelled as dynamical processes on complex networks, such as gene regulatory networks, social networks, and ecological networks [Newman, 2018]. Recovering the governing mechanism of such dynamical systems from observations is the first step towards understanding the global dynamical properties and eventually implementing interventions and controls. In a typical problem setup, a multivariate time series $\boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots, \boldsymbol{x}(t_N)$ is observed from a network dynamical system. From it, we aim to inductively infer a coupled dynamical system to model $\boldsymbol{x}(t)$. When the observations are evenly spaced, various solutions exist (e.g., [Takens, 1981; Brunton *et al.*, 2016]). But as many complex systems generate unevenly spaced observations, recovering the dynamic remains a challenge.

Many researchers exploited an encoder-decoder structure to recover coupled flows from time series data [Zang and Wang, 2020; Yan *et al.*, 2021; Wang *et al.*, 2022; Guo *et al.*, 2022; Hwang *et al.*, 2021]. Among these, the most representative model is NDCN [Zang and Wang, 2020]. As shown in Fig 1, instead of estimating a flow Φ from the original observations $\boldsymbol{x}$, NDCN seeks to simultaneously estimate an encoder f_e and a conjugate flow Φ' such that $\Phi'(t_i, f_e(\boldsymbol{x}_0)) \simeq (f_e \circ \Phi)(t_i, \boldsymbol{x}_0)$. As the new flow Φ' now lives in a higher-dimensional latent space, it potentially has a simpler form than Φ, in the same way that kernel methods allow linear algorithms to be used for nonlinear problems [Hofmann *et al.*, 2008]. However, this seemingly innocent use of an encoder can result in an incorrect dynamical model, in the sense defined by Section 3.1. In particular, we observe that a flow estimated by NDCN is often dynamically unstable (with the largest Lyapunov exponent being positive), even when the time series originates from a stable flow. We analyse the potential causes. By concluding that an encoder is unnecessary, we show that it is possible and more desirable to directly model the dynamics in its original state space, by proposing an effective and versatile embedding-free architecture.

In this paper, we make the following contributions.

- We expose a severe limitation of encoder-decoder-based models for neural dynamics on networks: superfluous latent dimensions induce incorrect flows. In particular, we show that some widely used models do not even produce a valid dynamical system. We demonstrate this phenomenon on various dynamics and networks.
- We propose an encoder-free architecture *Dy-Net Neural Dynamics* (DNND) to learn network dynamics from time series data. DNND goes beyond mere observation

*Corresponding authors.



(a) NDCN (b) DNND

Figure 1: Embedding-based neural dynamical model NDCN [Zang and Wang, 2020] (a) and our proposed *Dy-Net Neural Dynamics* (DNND) architecture (b). (a) NDCN simultaneously learns an encoder, a decoder, and the evolution rule. The encoder embeds an original variable x into a high-dimensional latent space. The evolution rule uses a predetermined linear coupling mechanism (the graph Laplacian) to approximate the unknown dynamical interactions among vertices. NDCN suffers from several problems, including poor generalisation beyond the training data (shown as the x-t curves diverging from the ground-truth curves from $t > 5$). (b) In contrast, DNND directly models the network dynamic on the original phase space. DNND uses two neural networks F_θ and G_θ to separately model the self-dynamic on nodes and the coupling along edges, explicitly allowing nonlinear coupling.

fitting and can reliably learn a model of the correct dynamical behaviours.

- We propose a loss warm-up procedure to address the numerical instability in training neural ODE on irregular time series data.
- We improve the existing evaluation protocol for neural dynamic models by including metrics aligned with common objectives of analysing nonlinear complex systems, including stability and limiting behaviours.

2 Background

2.1 Dynamics on Complex Networks

Consider an undirected network with n vertices. We represent the network with a symmetric adjacency matrix $\boldsymbol{A}$. An element A_{ij} in $\boldsymbol{A}$ is defined as

$$A_{ij} = \begin{cases} 1, & \text{if vertices } i \text{ and } j \text{ share an edge,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In this paper, we assume that the network contains no self-loop. Hence $A_{ii} = 0$ for any i.

Let $\mathcal{T} \triangleq \mathbb{R}_+$ (or more generally a monoid). We assume that each vertex i defines a continuous-time real-value state variable $x_i : \mathcal{T} \to \mathbb{R}$. And we consider a dynamical system defined by ordinary differential equations together with initial conditions:

$$\dot{x}_i(t) = \Psi_i(x_1(t), x_2(t), \ldots, x_n(t)) \tag{2}$$

$$x_i(0) = b_i, \tag{3}$$

where $t \in \mathcal{T}$ and $1 \leq i \leq n$. In this paper, we use $\dot{x}$ to denote $\frac{\mathrm{d}}{\mathrm{d}t}x$ (i.e., the velocity) and call $\Psi_i : \mathbb{R}^n \to \mathbb{R}$ a *velocity operator*. We assume that each velocity operator is smooth and globally Lipschitz. Also, we restrict Ψ_i to be autonomous (time-invariant).

Let $\boldsymbol{x}$ denote $(x_1, x_2, \ldots, x_n)^\top$. Then the evolution operators Ψ_i induce a *flow* $\Phi : \mathcal{T} \times \mathbb{R}^n \to \mathbb{R}^n$ as

$$\Phi(t, \boldsymbol{x}(0)) = \boldsymbol{x}(t) \tag{4}$$

$$= \int_0^t (\Psi_1(\boldsymbol{x}(\tau)), \ldots, \Psi_n(\boldsymbol{x}(\tau)))^\top \, \mathrm{d}\tau \tag{5}$$

The problem that concerns us in this paper is to recover Ψ_i from a multivariate time series:

$$\boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots, \boldsymbol{x}(t_N), \tag{6}$$

where $0 \leq t_1 < t_2 < \cdots < t_N$. Note that we do not assume that the time series is regularly spaced.

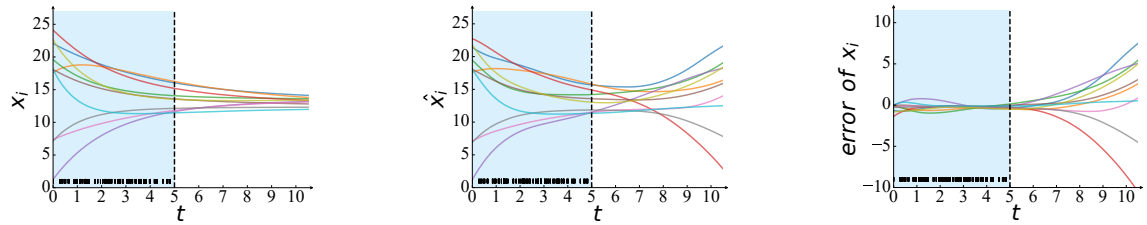
Clearly, in its most general form (2), it is an ill-defined problem to recover Φ from a single time series. If we review $\Delta t_i \triangleq t_{i+1} - t_i$ as the value of a roof function $r(\boldsymbol{x}(t_i))$, then Φ is a special flow [Fisher and Hasselblatt, 2019], and there are infinitely many valid special flows under time changes. Therefore, we need to introduce an additional inductive bias. Following [Zang and Wang, 2020], we assume a single operator $\Psi : \mathbb{R} \times \mathcal{X}_{\boldsymbol{A}} \to \mathbb{R}$ shared by all state variables:

$$\dot{x}_i(t) = \Psi(x_i(t), \{x_j(t) : A_{ij} = 1\}), \tag{7}$$

where $\mathcal{X}_{\boldsymbol{A}} \subset 2^{\mathbb{R}}$. In other words, $x_i(t)$ depends on itself and its neighbouring state variables, and Ψ is invariant to the permutation of the neighbouring vertices.

2.2 Encoder-Decoder Based Neural Dynamical Models

How do you model the operator Ψ in Eq (7)? Many neural dynamical models for complex networks (e.g., [Zang and Wang, 2020]) follow a common encoder-decoder architecture in graph neural networks [Kipf and Welling, 2016] and also in general neural ODEs [Lee and Parish, 2021]. In NDCN [Zang and Wang, 2020] (see Fig 1.(a)), the authors defined an encoder $f_E : \mathbb{R} \to \mathbb{R}^d$ and a decoder $f_D : \mathbb{R}^d \to \mathbb{R}$, where $d \in \mathbb{Z}_+$ is the embedding dimension, so that each initial value $x_i(0)$ is encoded jointly as d state variables in the



(a) true trajectories (b) estimated trajectories (c) prediction error

Figure 2: Trajectories of 10 vertices following the heat diffusion dynamics. (a) The ground-truth trajectories converge to the mean temperature. (b) An NDCN model was trained using 80 observations to the left of the dashed line (at times shown in the rug plot). The estimated trajectories fitted the training data well. But the trajectories diverge quickly outside the region with training observations. The trajectories certainly are inconsistent with how heat diffusion works.

vector $\boldsymbol{h}_i(0) = f_E(x(0))$. Then they define

$$\dot{\boldsymbol{H}}(t) = f_\theta(\boldsymbol{L}\boldsymbol{H}(t)), \tag{8}$$

where $\boldsymbol{L}$ is the graph Laplacian, $\boldsymbol{H}(t)$ is $(\boldsymbol{h}_1(t), \boldsymbol{h}_2(t), \ldots, \boldsymbol{h}_n(t))^\top$, and $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function. Clearly, the use of graph Laplacian for aggregation is a convenient choice. It restricts the type of dynamical coupling among neighbouring vertices.

Training an NDCN model involves jointly fitting f_E, f_θ, f_D to minimise the empirical risk on the observations. It is not different from a standard supervised training setup. In particular, no explicit requirement is imposed on the dynamical behaviours of the fitted flow.

3 Issues with Encoder-Decoder Based Neural Dynamical Models

In this section, we show that encoding the scalar state variable $x(t)$ as a latent vector $\boldsymbol{h}(t)$ leads to overfitting and spurious evolution operators. We use a simulated example to demonstrate the problems and then analyse potential causes in Section 3.3. Later in Section 4, we will propose a simple yet effective solution to address these problems.

3.1 Three Tests for Neural Dynamical Models

The universal approximation theorem [Hornik *et al.*, 1989] allows us to fit a neural network to approximate any time series. In NDCN, if we keep increasing the embedding dimension d, we can always fit a given observation time series. But for any fitted neural dynamical model to be useful, we expect the induced flow to have the correct dynamical properties, which we aim to capture with the following three tests.

Extrapolation beyond training data. To fit a neural dynamical model, we rely on observations at finite time points $t_1, t_2, \ldots, t_N$. These observations are often collected during a short-term transient period close to an initial condition. In many studies of NODEs, a test sample from a time period overlapping or close to the training period is used. This is consistent with the standard supervised setting, where data is assumed to be independent and identically distributed (iid) or nearly iid. However, in dynamical modelling, more important is the long-term behaviour of the fitted model. Therefore, the model needs to make accurate predictions on long-term horizons. So our first test is *whether the estimated model can make accurate long-term predictions beyond the observation period of the training data.*

Same fixed points or invariant sets. One may argue that in dynamic system modelling, what matters is the correct final state, instead of accurate predictions of intermediate state values. Related to the previous test is *whether the fitted model needs to have the same collection of fixed points (or invariant sets) as the true model.* If the answer is yes, two corresponding fixed points need to have the same dynamical stability. In this paper, we check the stability of a flow by the largest Lyapunov exponent [Wolf *et al.*, 1985].

Neural network as a legal flow. The final test is the most important but is often overlooked: *whether the fitted neural dynamical model induces a legal vector field for a flow*. In particular, the model should induce a state transition function that is a monoid action. This means that if Φ is a flow, then

$$\Phi(0, \boldsymbol{x}) = \boldsymbol{x} \tag{9}$$

$$\Phi\left(t_2, \Phi\left(t_1, \boldsymbol{x}\right)\right) = \Phi\left(t_2 + t_1, \boldsymbol{x}\right), \tag{10}$$

for any $t_1, t_2 \in \mathcal{T}$. In other words, the trajectory $\boldsymbol{x}(t)$ cannot branch out at any time t.

3.2 How NDCN Fares in These Tests

We demonstrate the problems with embedding-based neural dynamical models using an example from [Zang and Wang, 2020]. In this example, NDCN is shown to outperform various RNN+GNN baselines, in short-horizon extrapolation tasks (see Table 3 of [Zang and Wang, 2020]).

Heat diffusion on a frid network. Following [Zang and Wang, 2020], we simulate a vector time series using a heat equation defined on a grid network. We consider a network with 400 vertices arranged in a square 2d lattice. Let $\{v(k,l) : 1 \le k, l \le 20\}$ be the vertices. Two vertices $v(k,l)$ and $v(k',l')$ are connected if and only if $|k - k'| \le 1$ and $|l - l'| \le 1$. In other words, this is a modified version of the 8-regular graph with 2d boundaries. On this network, a flow is defined by the heat equation. For each node i, we have:

$$\dot{x}_i(t) = -\sum_{j=1}^{N} A_{ij}\alpha(x_i - x_j), \tag{11}$$

where α is a diffusive constant. Also $x_i(0)$ is set with a random value from $[0, 25]$. Note that in theory, the above

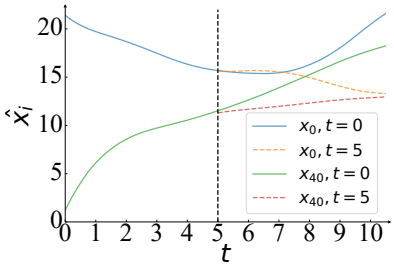


Figure 3: NDCN does not define a proper dynamical system as $\Phi(t_2, \Phi(t_1, \boldsymbol{x})) \neq \Phi(t_2 + t_1, \boldsymbol{x})$, violating the monoid-action requirement. This is shown as one trajectory branching into two trajectories (dashed vs solid).

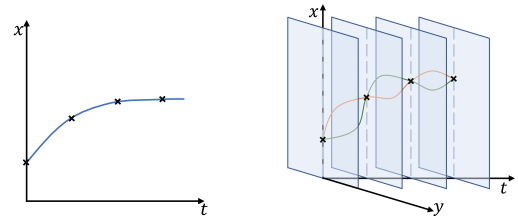


Figure 4: Additional embedding dimensions reduce the influence of observation data on the model fit. Increased hypothesis space leads to overly complex and unstable models, as suggested by Occam's razor [Blumer *et al.*, 1987].

coupling operator can be perfectly modelled by the graph Laplacian. From the above flow, we created 80 irregularly spaced samples with $t \in [0, 5]$, and fitted an NDCN model using the code provided by the authors on Git Hub (https://github.com/calvin-zcx/ndcn).

Failing to extrapolate. Fig 2 shows the true trajectories of 10 vertices and the corresponding trajectories recovered from the NDCN model. The NDCN model performed well in the neighbourhood of the training observations (to the left of the dashed line). But it has almost poor generalisation capacity outside that region. Fig 2(c) shows that the prediction error grows as we move away from the training data.

Missing the fixed point. The heat flow has a simple fixed point: eventually, every x_i converges to the average temperature of the network. Fig 2b shows that the NDCN model fails to capture this very stable fixed point, as all state variables actually diverge over time in the fitted model. We measured the largest Lyapunov exponent of the true flow and the estimated flow. While the true value is -35.83, indicating a stable flow, the value of the estimated flow is 9.36, a positive value that implies instability. The same contradiction can be revealed by a 2D visualisation of a vertex-wise phase plot. Fig 8 shows that the estimated flow initially approached the average temperature but eventually diverged from the correct fixed point.

Failing to produce a flow. From the estimated flow $\hat{\Phi}$, we produce two time series. One from $t = 0$ with the initial condition $x_i(0)$ and another from $t = 5$ with the initial condition $\hat{\Phi}(5, x_i(0))$. If $\hat{\Phi}$ is a legal flow, then these two time series should coincide from $t \geq 5$. Figure 3 shows the time series for two vertices. Clearly resetting the initial condition led to a completely different time series.

3.3 What Went Wrong?

The above example shows that a neural dynamical model can fit an observation time series well in the embedding space, but at the same time fails to produce the correct flow. Embedding has been used to reconstruct flows from regularly spaced time series [Takens, 1981]. Also, it has been shown that any homeomorphism $h : \mathcal{X} \to \mathcal{X}, \mathcal{X} \subset \mathbb{R}^p$ in a p-dimensional Euclidean space can be approximated by a Neural ODE $\hat{\Phi} : \mathcal{T} \times \mathbb{R}^{2p} \to \mathbb{R}^{2p}$ operating on a 2p-dimensional Euclidean space [Zhang *et al.*, 2020]. But then why did the embedding-based model fail to recover the heat dynamics?

Encoder/Decoder or Homeomorphism?

First, we limit our definition of a *correct* flow. Following [Fisher and Hasselblatt, 2019], a flow $\Phi = \{\phi(t, x)\}$ on X and another flow $\Psi = \{\psi(t, y)\}$ on Y are *topological equivalent* if there is a homeomorphism $h : X \to Y$ such that

$$h(\phi(t, x)) = \psi(t, h(x)). \tag{12}$$

If we accept the notion of "correctness", then in Fig 1(a) the encoder function f_E and the decoder function f_D have to satisfy some conditions.

First, f_E and f_D need to be inverse to each other, as they represent h and h^{-1}. This implies that f_E and f_D should not be implemented using two independent neural networks. Secondly, both f_E and f_D must be one-to-one functions (injections). This implies that the encoder/decoder neural network has to be reversible. In particular, it cannot contain pooling layers or a RELU activation function. However, the encoders and decoders used in most neural dynamical models do not meet these two conditions. In [Zang and Wang, 2020], f_D and f_E are parametrised using two separate neural networks, just as in a standard GNN.

Curse or Blessing of Dimensionality?

Another problem with embedding is the increased dimensionality of an inverse problem. As in NDCN, the encoder maps a scalar value to a vector of dimension d. So instead of fitting a shared scalar field as in Eq (7), we need to learn the d-dimension vector field. At the same time, we have only one time series of dimension n as the training data. When d is too large relative to n, overfitting is likely to occur. This is illustrated in Fig 4.

One may argue that embedding is routinely used by neural ODEs in classification/regression tasks [Chen *et al.*, 2018; Chamberlain *et al.*, 2021], and there appear to be no issues. In classification/regression tasks, neural ODEs produce a flow Φ and a return time T such that $\Phi(T, \cdot)$ serves as an approximator of the homeomorphism between inputs and label encoding. In such a scenario, it is well established that lifting the dimension of the input features enlarges the set of valid evolution functions, making a "difficult" homeomorphism easier to fit [Dupont *et al.*, 2019]. However, it is unnecessary to consider whether the flow Φ itself is "correct" or not. With

that said, superfluous dimensions may still lead to unstable fit [Massaroli *et al.*, 2020].

In NDCN, the additional embedding dimension does provide additional expressive power that compensates for the limitation of the graph Laplacian aggregation function. But we argue that if the graph Laplacian is too restrictive for the task, we should use a more general aggregation function.

4 Dy-Net Neural Dynamics

In the previous section, we have seen that an embedding-based neural dynamical model can learn a spurious flow that fits the training data well, but is qualitatively different from the true dynamic in terms of long-term properties.

If it is so difficult to learn an encoder/decoder pair that guarantees topological conjugacy, do we need latent embedding in the first place? In this section, we show the possibility of constructing neural networks that directly model the flow in its original state space.

4.1 Replacing Graph Laplacian with Permutation Invariance Aggregation

If we build a neural network to work directly on the original state space, one challenge is the limited expressive power of the graph Laplacian. The graph Laplacian and its variants play a central role in graph convolution and graph neural networks [Kipf and Welling, 2016; Veličković *et al.*, 2017]. However, the graph Laplacian is a linear aggregation operator that cannot handle potentially nonlinear coupling in network dynamics. In other words, to avoid encoders and decoders, we also have to abandon the graph Laplacian and seek a more general aggregation scheme. We propose a model based on the separation of self-dynamic and coupling operators.

4.2 Dy-Net Neural Dynamics

Our first approximation is to replace $\Psi(x_i(t), \{x_j(t) : A_{ij} = 1\})$ with the sum of a node operator and an edge operator:

$$\dot{x}_i(t) = \Psi_N(x_i(t)) + \Psi_E(\{x_j(t) : A_{ij} = 1\}). \tag{13}$$

By Theorem 2 in [Zaheer *et al.*, 2017], $\Psi_E(\{x_j(t) : A_{ij} = 1\}$ can be represented as $\rho\left(\sum_{j:A_{ij}=1} \phi(x_j)\right)$, where ρ and ϕ are two scalar transformations.

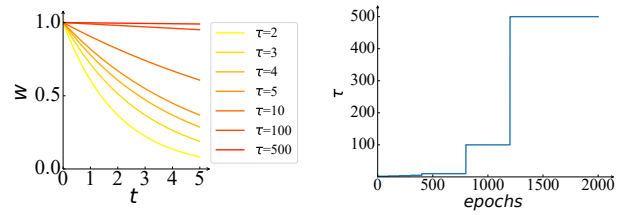
Inspired by the pairwise coupling feature in [Chen *et al.*, 2014], we further assume that ρ is additive and that each x_j can have a non-linear coupling effect on x_i governed by a shared term $\Psi_e(x_i(t), x_j(t))$. This leads to the following sum decomposition of the operator Ψ:

$$\dot{x}_i(t) = \Psi_N(x_i(t)) + \sum_{A_{ij}=1} \Psi_e(x_i(t), x_j(t))). \tag{14}$$

This turns out to be the same vector-field structure that appears in [Barzel and Barabási, 2013; Barzel *et al.*, 2015; Gao *et al.*, 2016].

Note that Eq (14) is more general than the graph Laplacian aggregation in Eq (11). It is also more general than the three-term structure in [Barzel *et al.*, 2015]:

$$\dot{x}_i(t) = M_0(x_i(t)) + \sum_{A_{ij}=1} M_1(x_i(t))M_2(x_j(t))). \tag{15}$$



(a) weight $w_k(t)$ at different temperature τ_k (b) temperature τ_k at different epoch k

Figure 5: Loss warm-up schedule for learning F and G. (a) At the early stage of the training process, the contribution of the data points far away from the initial time (t being large) are dampened to stabilise the training. (b) A temperature variable τ controls how the dampening is relaxed with the number of epochs. The temperature increases every 400 epochs.

Eq (14) suggests that we can use two separate neural networks F_θ and G_θ to parametrise Ψ_N and Ψ_e (see Fig. 1). This is in contrast to the single-neural-network solution in NDCN. Compared with an encoder-decoder-based model, a model specified by Eq (14) offers important benefits: The neural networks for Ψ_N and Ψ_e are independent of $\boldsymbol{A}$. That means that these two neural networks F_θ, G_θ can be trained even when the adjacency matrix $\boldsymbol{A}$ changes within a time series. Dynamical network rewiring is common in many applications, such as biological and social systems. Because of this decoupling of Ψ_e from $\boldsymbol{A}$, we call our two-neural-network model *Dy-Net Neural Dynamics* (DNND).

Fig 1(b) shows the architecture of DNND, which renders the following flow on networks:

$$\dot{x}_i(t) = F_\theta\left(x_i(t)\right) + \sum_{j=1}^{N} A_{ij} G_\theta\left(x_i(t), x_j(t)\right), \tag{16}$$

for all $1 \leq i \leq n$. Clearly $x_i(t)$ is always a flow by definition.

Affine regularisation. It is well known that neural ODE training can be numerically challenging due to approximation errors in numerical integration procedures and stiffness in the system [Kim *et al.*, 2021]. To alleviate the challenge of unstable model training, we model F_θ and G_θ as

$$F_\theta(x) = c_{f0} + c_{f1} \cdot x + f_\theta(x) \tag{17}$$

$$G_\theta(x_1, x_2) = c_{g0} + c_{g11} \cdot x_1 + c_{g12} \cdot x_2 + g_\theta(x_1, x_2) \tag{18}$$

where $c_{f0}, c_{f1}, c_{g0}, c_{g11}, c_{g12}$ are scalar parameters to capture the dominating linear terms, and f_θ and g_θ are MLPs to capture higher-order nonlinear terms. This is similar to the set-up of LassoNet [Lemhadri *et al.*, 2021]. It introduces the inductive bias of preferring affine mappings. It allows us to regularise f_θ and g_θ to promote sparse and interpretable models. Note that the graph Laplacian is now a special case with only two nonzero terms $c_{g11} = 1$ and $c_{g12} = -1$.

4.3 Robust Fitting of Two Neural Networks

Given a time series $D = \boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots, \boldsymbol{x}(t_N)$, the learning problem is to find $F_\theta : \mathbb{R} \to \mathbb{R}$ and $G_\theta : \mathbb{R}^2 \to \mathbb{R}$ that

dynamic	evolution rule for x_i
Heat	$\dot{x}_i = -0.1 \sum_j A_{ij}(x_i - x_j)$
Biochemical	$\dot{x}_i = 1 - 0.1x_i - 0.01 \sum_j A_{ij} x_i x_j$
BirthDeath	$\dot{x}_i = -0.1x_i^2 + 0.2 \sum_j A_{ij} x_j$

Table 1: Dynamical systems for generating observation time-series. More background on these formulas can be found in [Barzel and Barabási, 2013].

minimise the following loss:

$$\mathcal{L} = \mathcal{R}(\hat{\boldsymbol{x}}(t), D). \tag{19}$$

To reduce the impact of numerical error accumulation from long-range integration [Milani and Nobili, 1987], we devise a warm-up schedule to adapt the loss function dynamically. At the early stage of training, the model parameters are strongly affected by time-series data points closer to the initial value, which results in smaller integration errors. More specifically, we adopt a weighting schedule for the loss function with evolving weight function w_k at epoch k as follows:

$$w_k(t) = e^{-t/\tau_k}, \tag{20}$$

where τ_k monotonically increases with the number of training epochs k. So the loss function evolves as follows:

$$\mathcal{L}_k = \sum_{y_t \in D} \mathcal{R}(\hat{\boldsymbol{x}}(t), y_t) \cdot w_k(t) \tag{21}$$

5 Experiments

We use several approaches to recover evolution operators from time series and then compare the quality of the recovered models. We closely follow [Zang and Wang, 2020] to set up the ground-truth models and to simulate the observation data. In this setup, a ground-truth model is a coupled system of ODEs defined on a random network. From initial conditions, we simulate a time series as the observation data. Our goal is to recover a neural-network model for the evolution operators in the ground-truth ODE system.

5.1 Ground-truth Dynamical Models and Observation Data

We considered an extensive set of linear and nonlinear ODEs (Table 1) on different complex networks, including a fixed grid network from [Zang and Wang, 2020] and random networks based on the Erdós and Rényi (ER) model [Gilbert, 1959], the Albert-Barabási (BA) model [Barabási and Albert, 1999], the Watts-Strogatz (WS) model [Watts and Strogatz, 1998], and the LFR model [Fortunato, 2010; Lancichinetti *et al.*, 2008].

Training Time Series. We follow the setup in [Zang and Wang, 2020] and randomly sample 80 (irregularly spaced) times $0 \leq t_1 < t_2 < \cdots < t_{80} \leq 5$ for training. The initial conditions $\boldsymbol{x}(0)$ of the dynamical variables on vertices are set with random values from $[0, 25]$. We use the Runge–Kutta–Fehlberg (RKF45) solver to generate $\boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots, \boldsymbol{x}(t_{80})$ as the training time series.

dynamics	time	NDCN	DNND
heat	0-5	1.3±0.4	**0.3±0.1**
	5-6	1.5±0.4	**0.5±0.3**
	40-50	322.4±148.0	**11.9±1.8**
biochemical	0-5	6.2±0.8	**1.3±0.2**
	5-6	12.5±1.8	**1.0±0.1**
	40-50	27142.4±29581.0	**46.0±11.9**
birthdeath	0-5	1.4±0.3	**0.7±0.3**
	5-6	1.5±0.2	**0.3±0.1**
	40-50	1673.4±900.8	**0.4±0.1**

Table 2: Forecast MAPE (%) on the grid network. Models are trained on 80 observations from $[0, 5]$ and evaluated at random times in three periods: interpolation [0-5], short-term [5-6], and long-term [40-50].

5.2 Generalisation beyond the Training Data

Table 2 compares the prediction performance from the short-term prediction to the long-term prediction. One can see that NDCN performs reasonably well near the training data, but its long-term prediction is unreliable, with the mean absolute percentage error (MAPE) even exceeding 100%. In contrast, DNND produces accurate predictions, even well beyond the region of training data. The pattern persists on other network topologies (see more results in the supplementary material).

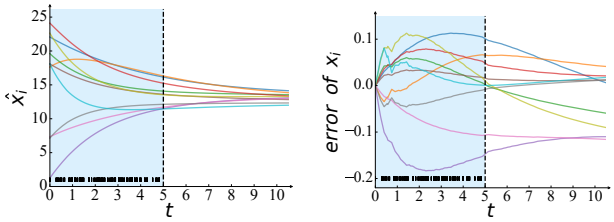
Fig. 7 shows time series on 50 vertices. NDCN produces time series with values far exceeding the physically reasonable values.

5.3 Recovery of Dynamic

The overfitting of NDCN is also revealed in the largest Lyapunov exponent of the fitting model. Table 3 shows that all dynamics are stable with the grid network, with the negative Lyapunov exponents. The DNND estimates are similar to the true values. In contrast, NDCN produces positive Lyapunov exponents, suggesting unstable dynamics.

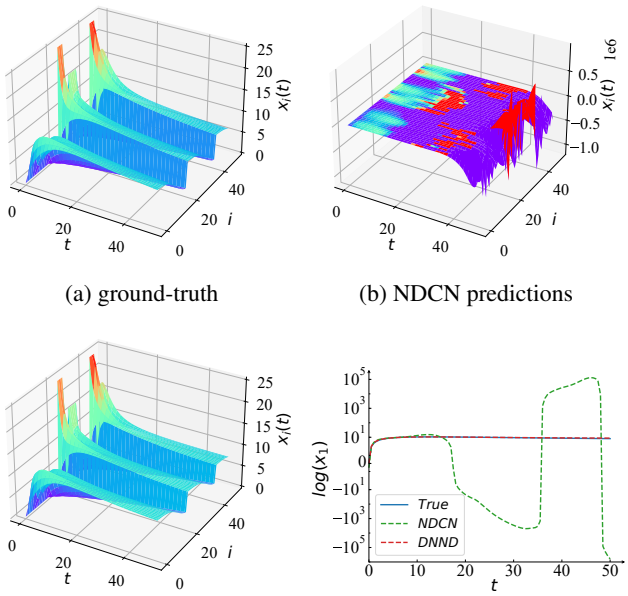
we can also see the fixed point, if any, of the fitted model using a projection into the 2D space spanned by x_i and $\dot{x}_i$. Fig. 8 shows that DNND converges to the correct fixed point. In contrast, NDCN does not converge at all.

Recovering Governing Evolution Function. As shown in [Murphy *et al.*, 2022], predictability and reconstructability



(a) estimated trajectories (b) estimation error

Figure 6: Time series estimated by DNND and the estimation error on the same dataset in Fig 2. The estimation error is bounded. (Note that the y axis here has a different range.)



(a) ground-truth (b) NDCN predictions

(c) DNND predictions (d) one vertex

Figure 7: Predicted time series on 50 vertices. NDCN predictions are completely wrong in the long term while DNND predictions are consistent with the ground truth. Note in (b) the vertical axis has a wider range. (d) Time series for node x_1; symmetric log plot [Danz *et al.*, 2021] was used due to the large difference in scales.

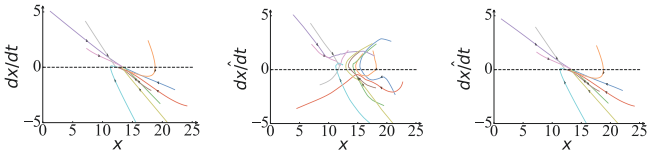
dynamics	true value	NDCN	DNND
Heat	-35.83	9.36	-34.97
Biochemical	-9.34	18.94	-10.23
Birthdeath	-92.07	15.70	-87.52

Table 3: The largest Lyapunov exponent of the ground-truth models and the estimated models. Positive values indicate unstable (potentially chaotic) dynamics.

are independent and sometimes incompatible problem properties. A time series with less variability is easier to predict but provides less information for recovering the evolution function. Fig. 9 shows that from the two subnetworks in DNND, we can reasonably recover the vector field in terms of self-dynamic and coupling effect: Ψ_N and Ψ_e in Eq. (14).

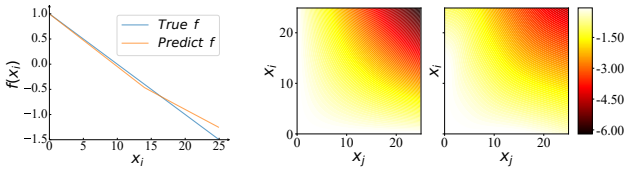
Note that NDCN's vector field is defined in a latent space and it is impossible to recover the evolution function in terms of the original state variables. Unlike NDCN, which models the governing function on latent variables, DNND directly captures the governing function of the dynamic in terms of the original physical measurements. From the F_θ-network and G_θ-network, we can recover the input-output dependency. It is another advantage of our embedding-free approach.

Furthermore, if Ψ_N and Ψ_e have analytical forms in terms of basis functions, then symbolic regression can potentially be applied to recover the exact algebraic forms of the two functions. We show some results in the Appendix.



(a) ground truth (b) NDCN (c) DNND

Figure 8: 2D projection of trajectories from 9 random nodes in the grid network. The dynamic is the heat flow. The ground-truth trajectories converge to the only fixed point of the system, with every node having the average heat. (b) NDCN trajectories are not converging to a fixed point.



(a) (b)

Figure 9: Functions recovered by DNND. (a) true and estimated graphs of F_θ (b) heatmaps for the value of $G_\theta(x_i, x_j)$, with the true one on the left and the estimated one on the right. The true dynamic is the biochemical dynamic in Table 1.

6 Conclusion

We show that blindly applying an encoder-decoder structure, without a sound principle such as topological conjugacy, is harmful to correct recovery of a dynamic flow from observations. Through our proposed *Dy-Net Neural Dynamics* (DNND) model, we show that state embedding is unnecessary for a broad class of flow-recovery problems. Empirically, DNND reliably recovers a wide range of dynamical systems on different network topologies.

As DNND models the vector field in the original state space, the fitted model can serve as a basis for discovering analytical, hence interpretable, differential equations using techniques such as symbolic regression [Brunton *et al.*, 2016]. We presented some early results in the Appendix.

Theoretically, it would be interesting to characterise the learnability of dynamical flows, up to topological equivalence, from irregularly spaced observations. This is related to finding a suspension flow under a roof function [Fisher and Hasselblatt, 2019]. But as we have access to only samples of the homeomorphism, it is a machine learning problem. One particularly interesting question worth exploring is how to parametrise time changes ($\frac{\partial}{\partial t}$) given a finite sample from an unknown roof function.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant Nos. 2021ZD0112501 and 2021ZD0112502; the National Natural Science Foundation of China under Grant Nos. U22A2098, 62172185, 62206105 and 62202200; Jilin Province Capital Construction Fund Industry Technology Research and Development Project No.

2022C047-1; Changchun Key Scientific and Technological Research and Development Project under Grant No. 21ZGN30; Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education Open Research Project No. 93K172021K05.

References

[Barabási and Albert, 1999] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[Barzel and Barabási, 2013] Baruch Barzel and Albert-László Barabási. Universality in network dynamics. *Nature physics*, 9(10):673–681, 2013.

[Barzel *et al.*, 2015] Baruch Barzel, Yang-Yu Liu, and Albert-László Barabási. Constructing minimal models for complex system dynamics. *Nature communications*, 6(1):1–8, 2015.

[Blumer *et al.*, 1987] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

[Brunton *et al.*, 2016] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[Chamberlain *et al.*, 2021] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.

[Chen *et al.*, 2014] Xu Chen, Xiuyuan Cheng, and Stéphane Mallat. Unsupervised deep haar scattering on graphs. *Advances in Neural Information Processing Systems*, 27, 2014.

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[Danz *et al.*, 2021] Thomas Danz, Till Domröse, and Claus Ropers. Ultrafast nanoimaging of the order parameter in a structural phase transition. *Science*, 371(6527):371–374, 2021.

[Dupont *et al.*, 2019] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in Neural Information Processing Systems*, 32, 2019.

[Fisher and Hasselblatt, 2019] Todd Fisher and Boris Hasselblatt. *Hyperbolic flows*. EMS Press, 2019.

[Fortunato, 2010] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[Gao *et al.*, 2016] Jianxi Gao, Baruch Barzel, and Albert-László Barabási. Universal resilience patterns in complex networks. *Nature*, 530(7590):307–312, 2016.

[Gilbert, 1959] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[Guo *et al.*, 2022] Jia-Nan Guo, Peiyan Zhang, C. Li, Xing Xie, Yan Zhang, and Sunghun Kim. Evolutionary preference learning via graph nested gru ode for session-based recommendation. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022.

[Hofmann *et al.*, 2008] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, 36(3):1171–1220, 2008.

[Hornik *et al.*, 1989] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[Hwang *et al.*, 2021] JeeHyun Hwang, Jeongwhan Choi, Hwan-Kyu Choi, Kookjin Lee, Dongeun Lee, and Noseong Park. Climate modeling with neural diffusion equations. *2021 IEEE International Conference on Data Mining (ICDM)*, pages 230–239, 2021.

[Kim *et al.*, 2021] Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9):093122, 2021.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Lancichinetti *et al.*, 2008] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[Lee and Parish, 2021] Kookjin Lee and Eric J Parish. Parameterized neural ordinary differential equations: Applications to computational physics problems. *Proceedings of the Royal Society A*, 477(2253):20210162, 2021.

[Lemhadri *et al.*, 2021] Ismael Lemhadri, Feng Ruan, and Rob Tibshirani. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pages 10–18. PMLR, 2021.

[Massaroli *et al.*, 2020] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *Advances in Neural Information Processing Systems*, 33:3952–3963, 2020.

[Milani and Nobili, 1987] Andrea Milani and Anna Maria Nobili. Integration error over very long time spans. *Celestial mechanics*, 43:1–34, 1987.

[Murphy *et al.*, 2022] Charles Murphy, Vincent Thibeault, Antoine Allard, and Patrick Desrosiers. Duality between predictability and reconstructability in complex systems. *arXiv preprint arXiv:2206.04000*, 2022.

[Newman, 2018] Mark Newman. *Networks*. Oxford university press, 2018.

[Takens, 1981] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang *et al.*, 2022] Zhongyang Wang, Junchang Xin, Qi Chen, Zhiqiong Wang, and Xinlei Wang. Ndcn-brain: An extensible dynamic functional brain network model. *Diagnostics*, 12, 2022.

[Watts and Strogatz, 1998] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.

[Wolf *et al.*, 1985] Alan Wolf, Jack B Swift, Harry L Swinney, and John A Vastano. Determining lyapunov exponents from a time series. *Physica D: nonlinear phenomena*, 16(3):285–317, 1985.

[Yan *et al.*, 2021] Xu Yan, Xiaoliang Fan, Peizhen Yang, Zonghan Wu, Shirui Pan, Longbiao Chen, Yu Zang, and Cheng Wang. Contig: Continuous representation learning on temporal interaction graphs. *ArXiv*, abs/2110.06088, 2021.

[Zaheer *et al.*, 2017] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[Zang and Wang, 2020] Chengxi Zang and Fei Wang. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 892–902, 2020.

[Zhang *et al.*, 2020] Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation capabilities of neural odes and invertible residual networks. In *International Conference on Machine Learning*, pages 11086–11095. PMLR, 2020.