SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Supplementary Information of Learning Continuous Network Emerging Dynamics from Scarce Observations via Data-Adaptive Stochastic Processes

Jiaxu Cui[1,2], Qipeng Wang[1,2], Bingyi Sun[2,3], Jiming Liu[4] & Bo Yang[1,2*]

[1]*College of Computer Science and Technology, Jilin University, Changchun* 130012*, China;*
[2]*Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun* 130012*, China;*
[3]*Public Computer Education and Research Center, Jilin University, Changchun* 130012*, China;*
[4]*Department of Computer Science, Hong Kong Baptist University, Hong Kong* 999077*, China*

Contents

Appendix A Model Details . . . 2
Appendix A.1 Details of the Neural ODE Processes for Network Dynamics . . . 2
Appendix A.1.1 *Design of the neural networks in the NDP4ND* . . . 3
Appendix A.1.2 *The derivation of training loss* . . . 3
Appendix A.1.3 *The derivation of predictive distribution* . . . 4
Appendix A.2 Proofs . . . 5
Appendix A.3 Computational complexity analysis. . . . 5
Appendix B Experimental Details . . . 6
Appendix B.1 Construction for training and testing sets . . . 6
Appendix B.2 Network dynamics scenarios . . . 6
Appendix B.2.1 *Mutualistic interacting dynamics* . . . 6
Appendix B.2.2 *Second-order phototaxis dynamics* . . . 7
Appendix B.2.3 *Brain dynamics* . . . 7
Appendix B.2.4 *Compartment models in epidemiology* . . . 8
Appendix B.2.5 *Empirical systems* . . . 8
Appendix B.2.6 *Settings for sparse and noisy observations* . . . 9
Appendix B.3 Baselines . . . 9
Appendix B.4 Metrics . . . 9
Appendix B.5 Experimental setup . . . 10
Appendix C Addtional results . . . 11
Appendix C.1 Tables . . . 11
Appendix C.2 Figures . . . 12

* Corresponding author (email: ybo@jlu.edu.cn)

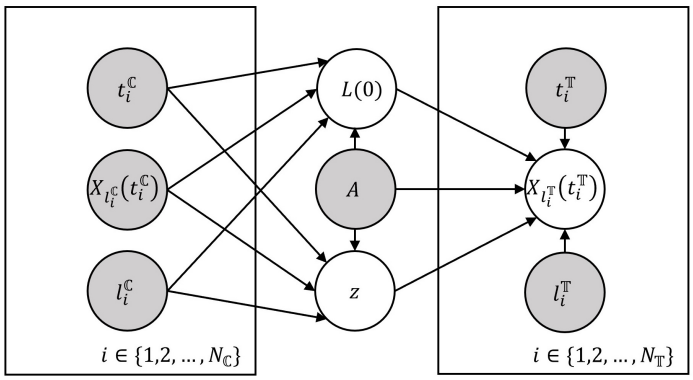

Figure A1 The probabilistic graphical model of the generative process behind the NDP4ND.

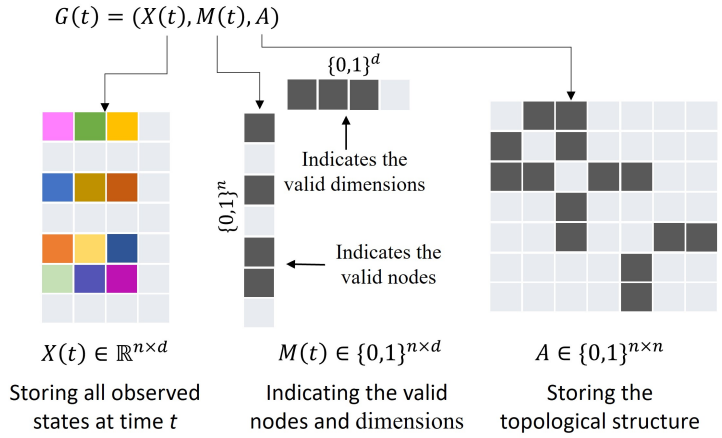

Figure A2 An illustration for $G(t) = (X(t), M(t), A)$. $X(t) \in \mathbb{R}^{n\times d}$ stores all observed states at time t and the mask $M(t)$ has the same shape as $X(t)$, indicating which nodes have been observed (its value on the corresponding position is set to 1, otherwise it is 0).

Appendix A Model Details

We propose the Neural ODE Processes for Network Dynamics (NDP4ND) to represent a spatio-temporal trajectory function space of network dynamics for covering abundant network dynamics instances. The probabilistic graphical model of the generative process behind the NDP4ND is shown in Figure A1. We first introduce the details of the NDP4ND, including the complete description of its computational diagram, shown in Figure 2 in the main text, and the architecture design of neural networks. The proofs of the propositions in the main text will be provided later.

Appendix A.1 Details of the Neural ODE Processes for Network Dynamics

The overall computational diagram consists of the following five main steps (corresponding to Figure 2 in the main text):

a, Let the observations ($\mathcal{D}_{\mathbb{C}}$) that capture the empirical topological structure and spatio-temporal behaviors be as the input of the NDP4ND.

b, Observations $\mathcal{D}_{\mathbb{C}}$ are first integrated into graphs based on different timestamps t, i.e., $G(t) = (X(t), M(t), A)$. $X(t) \in \mathbb{R}^{n\times d}$ stores all observed states at time t and the mask $M(t)$ has the same shape as $X(t)$, indicating which nodes have been observed (its value on the corresponding position is set to 1, otherwise it is 0). Figure A2 provides an illustration for $G(t)$. Then, by using a neural network $\boldsymbol{\varphi}$, the observations can be encoded into the embedding (r) with considering the temporal and topological information. Based on the embedding (r) of observations, a neural network $\boldsymbol{\rho}$ can be used to give the distribution of the global latent vector, i.e., $q(z|\mathcal{D}_{\mathbb{C}}, A)$. The distribution is to sample specific global control variables z and introduces uncertainty for propagation equations of network dynamics in hidden spaces.

c, Meanwhile, based on observations, the distribution of the initial latent state ($q(L(0)|\mathcal{D}_{\mathbb{C}}, A)$) can be obtained through a neural network $\boldsymbol{e}$.

d, Built on sampled global control variable z and deterministic representation of observations r, neural dynamics equation in hidden space is constructed via universal skeleton of network dynamics [1], i.e., $\dot{L}_{l_i^{\mathbb{T}}}(t) = \boldsymbol{S}(L_{l_i^{\mathbb{T}}}(t), \tilde{z}) + \sum_{j=1}^{n} A_{l_i^{\mathbb{T}}, l_j^{\mathbb{T}}} \boldsymbol{I}(L_{l_i^{\mathbb{T}}}(t), L_{l_j^{\mathbb{T}}}(t), \tilde{z})$, where $\tilde{z} = [z; r]$ and n is the system size. $\boldsymbol{S}$ and $\boldsymbol{I}$ are neural networks that characterize self dynamics and interaction dynamics, respectively.

e, Given the neural dynamics equation, topology A, initial latent state $L(0)$, test node $l_i^{\mathbb{T}}$, and test time $t_i^{\mathbb{T}}$, we can evolve the hidden state by solving the network ODE as $L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}) = L_{l_i^{\mathbb{T}}}(0) + \int_{t_0}^{t_i^{\mathbb{T}}} \left(\boldsymbol{S}(L_{l_i^{\mathbb{T}}}(t), \tilde{z}) + \sum_{j=1}^{n} A_{l_i^{\mathbb{T}}, l_j^{\mathbb{T}}} \boldsymbol{I}(L_{l_i^{\mathbb{T}}}(t), L_{l_j^{\mathbb{T}}}(t), \tilde{z}) \right) dt$. Here we use the adjoint method [2] to solve it to ensure that the model can perform backpropagation to learn parameters. Assuming that

output states are noisy, for a given $L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})$, we can decode it into the predictive state by $X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}) \sim p(X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})|t_i^{\mathbb{T}}, l_i^{\mathbb{T}}, z, L(0), A) = \mathcal{N}(\mu_{X_{l_i^{\mathbb{T}}}}(t_i^{\mathbb{T}}), \sigma^2_{X_{l_i^{\mathbb{T}}}}(t_i^{\mathbb{T}}))$, where $[\mu_{X_{l_i^{\mathbb{T}}}}(t_i^{\mathbb{T}}); \sigma^2_{X_{l_i^{\mathbb{T}}}}(t_i^{\mathbb{T}})] = \boldsymbol{d}(t_i^{\mathbb{T}}, L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}), \tilde{z})$ and $\boldsymbol{d}$ is as a neural network. Using $\tilde{z}$ as the input of $\boldsymbol{d}$ is equivalent to add an information flow path that skips the ODE process, i.e., a skip connection. This makes the NDP4ND at least as expressive as Neural Processes (NPs) [3] and Neural ODE Processes (NDPs) [4] in principle, also enabling it to model network dynamics that are not solely determined by some underlying ODEs.

Appendix A.1.1 *Design of the neural networks in the NDP4ND*

The neural networks involved in the model architecture include: $\boldsymbol{\varphi}$, $\boldsymbol{\rho}$, $\boldsymbol{e}$, $\boldsymbol{S}$, $\boldsymbol{I}$, and $\boldsymbol{d}$. We provide their specific implementations in the experiments below.

$\boldsymbol{\varphi} : [t_k; G(t_k)] \to r_k$.

[itemindent=1cm]$h_t = enc_t(t_k)$, where enc_t is a multilayer perceptron with 1 hidden layer and LeakyReLU activations. $h_{G(t_k)} = enc_G(G(t_k))$, enc_G has 2 graph attention (GAT) [5] layers with the head number of 8 and LeakyReLU activations. We choose GAT here due to its inductive learning power and excellent performance in experience. Note that the original states $X(t)$ are encoded before feeding enc_G, i.e., $h_{X(t)} = enc_X(X(t))$, where enc_X is a multilayer perceptron with 1 hidden layer and LeakyReLU activations. $r_k = enc_r([h_t; h_{G(t)}])$, where enc_r is a multilayer perceptron with 2 hidden layers and LeakyReLU activations.

To obtain a representation of the observations, we use an aggregation operation with permutation invariance to aggregate all r_k, i.e., $r = agg(\{r_k\}_{k=1}^K)$. We choose the element-wise mean as the aggregation operation.

$\boldsymbol{\rho} : r \to [\mu_z; \sigma_z^2]$, where μ_z and σ_z^2 are the mean and variance of the distribution $q(z|\mathcal{D}_{\mathbb{C}}, A)$.

[itemindent=1cm]$h_z = enc_z(r)$, where enc_z is a multilayer perceptron with 1 hidden layer and LeakyReLU activations. $\mu_z = enc_{\mu_z}(h_z)$, where enc_{μ_z} is a multilayer perceptron with 1 hidden layer. $\sigma_z = 0.1 + 0.9 \times \text{sigmoid}(enc_{\sigma_z}(h_z))$, where enc_{σ_z} is a multilayer perceptron with 1 hidden layer.

$\boldsymbol{e} : \mathcal{D}_{\mathbb{C}} \to [\mu_{L(0)}, \sigma^2_{L(0)}]$, where $\mu_{L(0)}$ and $\sigma^2_{L(0)}$ are the mean and variance of the distribution $q(L(0)|\mathcal{D}_{\mathbb{C}}, A)$. Although the distribution of $L(0)$ can also be obtained through the above fashion, we handle it in a simpler way when the initial states are always known.

[itemindent=1cm]$h_{L(0)} = enc_{L(0)}(X(0))$, where $X(0)$ is a shorthand for the initial states of all nodes and $enc_{L(0)}$ is a multilayer perceptron with 1 hidden layer and LeakyReLU activations. $\mu_{L(0)} = enc_{\mu_{L(0)}}(h_z)$, where $enc_{\mu_{L(0)}}$ is a multilayer perceptron with 1 hidden layer. $\sigma_{L(0)} = 0.1 + 0.9 \times \text{sigmoid}(enc_{\sigma_{L(0)}}(h_{L(0)}))$, where $enc_{\sigma_{L(0)}}$ is a multilayer perceptron with 1 hidden layer.

$\boldsymbol{S} : [L_{l_i^{\mathbb{T}}}(t); \tilde{z}] \to L^{self}_{l_i^{\mathbb{T}}}(t)$, where $\tilde{z} = [r; z]$ and $l_i^{\mathbb{T}}$ is the location of test node.

[itemindent=1cm]$L^{self}_{l_i^{\mathbb{T}}}(t) = S(L_{l_i^{\mathbb{T}}}(t), \tilde{z})$, where S is a multilayer perceptron with 2 hidden layers and LeakyReLU activations.

$\boldsymbol{I} : [L_{l_i^{\mathbb{T}}}(t); L_{l_j^{\mathbb{T}}}(t); \tilde{z}] \to L^{inter}_{l_i^{\mathbb{T}}, l_j^{\mathbb{T}}}(t)$, where $\tilde{z} = [r; z]$, $l_i^{\mathbb{T}}$ is the location of test node and $l_j^{\mathbb{T}}$ is the node that interacts with $l_i^{\mathbb{T}}$.

[itemindent=1cm]$L^{inter}_{l_i^{\mathbb{T}}, l_j^{\mathbb{T}}}(t) = I(L_{l_i^{\mathbb{T}}}(t), L_{l_j^{\mathbb{T}}}(t), \tilde{z})$, where I is a multilayer perceptron with 2 hidden layers and LeakyReLU activations.

$\boldsymbol{d} : [L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}); t_i^{\mathbb{T}}; \tilde{z}] \to [\mu_{X_{l_i^{\mathbb{T}}}}(t_i^{\mathbb{T}}); \sigma^2_{X_{l_i^{\mathbb{T}}}}(t_i^{\mathbb{T}})]$, where $\tilde{z} = [r; z]$, $l_i^{\mathbb{T}}$ and $t_i^{\mathbb{T}}$ are the test node and test time, respectively.

[itemindent=1cm]$L^{dec}_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}) = dec_L(L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}), t_i^{\mathbb{T}}, \tilde{z})$, where dec_L a multilayer perceptron with 2 hidden layers and LeakyReLU activations. $\mu_{X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})} = dec_{\mu_X}(L^{dec}_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}))$, where enc_{μ_X} is a multilayer perceptron with 1 hidden layer. $\sigma_{X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})} = 0.01 + 0.99 \times \text{softplus}(dec_{\sigma_X}(L^{dec}_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})))$, where, $\text{softplus}(a) = \log 1 + e^a$ and dec_{σ_X} is a multilayer perceptron with 1 hidden layer.

Although automatic machine learning technology, such as Bayesian optimization [6], can be used for fine-grained hyper-parameter optimization, we empirically set the hidden dimension of all multilayer perceptrons to 20, and the hidden dimension of graph attention layers to 16.

Appendix A.1.2 *The derivation of training loss*

Given a network A and a context set $\mathcal{D}_{\mathbb{C}}$, we revisit the generative model of the NDP4ND as

$$p_{X,z,L(0)} = p(z|\mathcal{D}_{\mathbb{C}}, A)p(L(0)|\mathcal{D}_{\mathbb{C}}, A)\prod_{i=1}^{N_{\mathbb{T}}} p(X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})|t_i^{\mathbb{T}}, l_i^{\mathbb{T}}, z, L(0), A),$$

where $p_{X,z,L(0)}$ is a shorthand for the generative model $p(X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}}), z, L(0)|t^{\mathbb{T}}_{1:N_{\mathbb{T}}}, l^{\mathbb{T}}_{1:N_{\mathbb{T}}}, \mathcal{D}_{\mathbb{C}}, A)$, $t^{\mathbb{T}}_{1:N_{\mathbb{T}}}$, $l^{\mathbb{T}}_{1:N_{\mathbb{T}}}$, and $X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}})$ are the target set, $L(0)$ and z denote the initial states in latent space and the global random vector that can control the network ODE, respectively.

Due to the embedded neural networks, the generative process is highly nonlinear, resulting in intractability to calculate the true posterior. We, thus, follow [3, 4, 7] and use an amortized variational inference to learn the parameters in the neural networks.

Given a network A, a context set $\mathcal{D}_{\mathbb{C}}$ and a target set $\mathcal{D}_{\mathbb{T}}$, where $\mathcal{D}_{\mathbb{C}} = \{(t_1^{\mathbb{C}}, l_1^{\mathbb{C}}, X_{l_1^{\mathbb{C}}}(t_1^{\mathbb{C}})), ..., (t_{N_{\mathbb{C}}}^{\mathbb{C}}, l_{N_{\mathbb{C}}}^{\mathbb{C}}, X_{l_{N_{\mathbb{C}}}^{\mathbb{C}}}(t_{N_{\mathbb{C}}}^{\mathbb{C}}))\}$ and $\mathcal{D}_{\mathbb{T}} = \{(t_1^{\mathbb{T}}, l_1^{\mathbb{T}}, X_{l_1^{\mathbb{T}}}(t_1^{\mathbb{T}})), ..., (t_{N_{\mathbb{T}}}^{\mathbb{T}}, l_{N_{\mathbb{T}}}^{\mathbb{T}}, X_{l_{N_{\mathbb{T}}}^{\mathbb{T}}}(t_{N_{\mathbb{T}}}^{\mathbb{T}}))\}$, the derivation process of the variational evidence lower-bound (ELBO) is as

follows

$$
\begin{aligned}
&\log p(X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}})|t^{\mathbb{T}}_{1:N_{\mathbb{T}}}, l^{\mathbb{T}}_{1:N_{\mathbb{T}}}, \mathcal{D}_{\mathbb{C}}, A)\\
=&\log \frac{p(z|\mathcal{D}_{\mathbb{C}}, A)p(L(0)|\mathcal{D}_{\mathbb{C}}, A)\prod_{i=1}^{N_{\mathbb{T}}} p(X_{l^{\mathbb{T}}_i}(t^{\mathbb{T}}_i)|t^{\mathbb{T}}_i, l^{\mathbb{T}}_i, z, L(0), A)}{p(z|\mathcal{D}_{\mathbb{T}}, A)p(L(0)|\mathcal{D}_{\mathbb{T}}, A)}\\
=&\mathbb{E}_{q(z|\mathcal{D}_{\mathbb{T}},A)q(L(0)|\mathcal{D}_{\mathbb{T}},A)}\Big[\sum_{i=1}^{N_{\mathbb{T}}}\log p(X_{l^{\mathbb{T}}_i}(t^{\mathbb{T}}_i)|t^{\mathbb{T}}_i, l^{\mathbb{T}}_i, z, L(0), A)\Big]\\
&- KL(q(z|\mathcal{D}_{\mathbb{T}}, A)\|q(z|\mathcal{D}_{\mathbb{C}}, A)) - KL(q(L(0)|\mathcal{D}_{\mathbb{T}}, A)\|q(L(0)|\mathcal{D}_{\mathbb{C}}, A))\\
&+ KL(q(z|\mathcal{D}_{\mathbb{T}}, A)\|q(z|\mathcal{D}_{\mathbb{T}}, A)) + KL(q(L(0)|\mathcal{D}_{\mathbb{T}}, A)\|q(L(0)|\mathcal{T}_{\mathbb{C}}, A))\\
\geqslant&\mathbb{E}_{q(z|\mathcal{D}_{\mathbb{T}},A)q(L(0)|\mathcal{D}_{\mathbb{T}},A)}\Big[\sum_{i=1}^{N_{\mathbb{T}}}\log p(X_{l^{\mathbb{T}}_i}(t^{\mathbb{T}}_i)|t^{\mathbb{T}}_i, l^{\mathbb{T}}_i, z, L(0), A)\Big]\\
&- KL(q(z|\mathcal{D}_{\mathbb{T}}, A)\|q(z|\mathcal{D}_{\mathbb{C}}, A)) - KL(q(L(0)|\mathcal{D}_{\mathbb{T}}, A)\|q(L(0)|\mathcal{D}_{\mathbb{C}}, A)).
\end{aligned}
$$

Since $L(0)$ is only related to $X(0)$ in our design, when the initial states are always known in our implementation, we have $q(L(0)|\mathcal{D}_{\mathbb{C}}, A) = q(L(0)|\mathcal{D}_{\mathbb{T}}, A)$. Thus, we can simplify an approximate lower bound ($\mathcal{ELBO}$) as

$$
\begin{aligned}
&\log p(X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}})|t^{\mathbb{T}}_{1:N_{\mathbb{T}}}, l^{\mathbb{T}}_{1:N_{\mathbb{T}}}, \mathcal{D}_{\mathbb{C}}, A)\\
&\geqslant \mathbb{E}_{q(z|\mathcal{D}_{\mathbb{T}},A)q(L(0)|\mathcal{D}_{\mathbb{T}},A)}\Big[\sum_{i=1}^{N_{\mathbb{T}}}\log p(X_{l^{\mathbb{T}}_i}(t^{\mathbb{T}}_i)|t^{\mathbb{T}}_i, l^{\mathbb{T}}_i, z, L(0), A)\Big]\\
&- \beta KL(q(z|\mathcal{D}_{\mathbb{T}}, A)\|q(z|\mathcal{D}_{\mathbb{C}}, A)),
\end{aligned}
$$

where β is a tunable parameter to relieve the KL vanishing.

To learn the distribution over trajectory functions, we train the model on a set of state observations generated from B different network dynamics with $A^{(1)}, ..., A^{(B)}$ topological structures. Also, we split the observations into B context sets $\mathcal{D}^{1:B}_{\mathbb{C}}$ and B target sets $\mathcal{D}^{1:B}_{\mathbb{T}}$. $\mathcal{D}^{b}_{\mathbb{C}}$ is usually a subset of $\mathcal{D}^{b}_{\mathbb{T}}$. By accumulating the approximate lower bounds ($\mathcal{ELBO}_1, \mathcal{ELBO}_2, ..., \mathcal{ELBO}_B$) of B network dynamics, the derived total training loss under all observations is as follows

$$
\begin{aligned}
\mathcal{L} =&\frac{1}{B}\sum_{b=1}^{B} -\mathcal{ELBO}_b\\
=&\frac{1}{B}\sum_{b=1}^{B}\Big[\mathbb{E}_{q(z|\mathcal{D}^b_{\mathbb{T}},A^{(b)})q(L(0)|\mathcal{D}^b_{\mathbb{T}},A)}\Big(\sum_{i=1}^{N^b_{\mathbb{T}}} -\log p(X_{l^{\mathbb{T}_b}_i}(t^{\mathbb{T}_b}_i)|t^{\mathbb{T}_b}_i, l^{\mathbb{T}_b}_i, z, L(0), A^{(b)})\Big)\\
&+ \beta KL(q(z|\mathcal{D}^b_{\mathbb{T}}, A^{(b)})\|q(z|\mathcal{D}^b_{\mathbb{C}}, A^{(b)}))\Big].
\end{aligned}
$$

We use the reparametrization trick and stochastic gradient descent to minimize the loss.

Appendix A.1.3 *The derivation of predictive distribution*

Given the learned NDP4ND, a network topology A, and a set of state observations $\mathcal{D}_{\mathbb{C}}$ from the task at hand, we obtain the predictive distribution of the states on any node $l^{\mathbb{T}}$ at any time $t^{\mathbb{T}}$ as

$$
p(X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A) = \int q(z|\mathcal{D}_{\mathbb{C}}, A)q(L(0)|\mathcal{D}_{\mathbb{C}}, A)p(X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, z, L(0), A)dzdL(0),
$$

where, $p(X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, z, L(0), A) = \mathcal{N}(\mu_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}), \sigma^2_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}))$. $\mu^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}})$ and $\sigma^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}})$ are outputs of decoder network $\boldsymbol{d}$.

We can use the Monte Carlo method to approximate the integral and obtain the first and second moments as follows

$$
\begin{aligned}
\mathbb{E}[X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A] &= \int \mathbb{E}[X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, z, L(0), A]q(z|\mathcal{D}_{\mathbb{C}}, A)q(L(0)|\mathcal{D}_{\mathbb{C}}, A)dzdL(0)\\
&= \int \mu_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}})q(z|\mathcal{D}_{\mathbb{C}}, A)q(L(0)|\mathcal{D}_{\mathbb{C}}, A)dzdL(0)\\
&\approx \frac{1}{J}\sum_{j=1}^{J}\mu^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}),
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}[X^2_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A] &= \int \mathbb{E}[X^2_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, z, L(0), A]q(z|\mathcal{D}_{\mathbb{C}}, A)q(L(0)|\mathcal{D}_{\mathbb{C}}, A)dzdL(0)\\
&= \int [(\sigma_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}))^2 + (\mu_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}))^2]q(z|\mathcal{D}_{\mathbb{C}}, A)q(L(0)|\mathcal{D}_{\mathbb{C}}, A)dzdL(0)\\
&\approx \frac{1}{J}\sum_{j=1}^{J}(\sigma^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}))^2 + (\mu^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}}))^2,
\end{aligned}
$$

where J is the sampling number for z and $L(0)$. We calculate $\mu^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}})$ and $\sigma^{(j)}_{X_{l^{\mathbb{T}}}}(t^{\mathbb{T}})$ by sampling $z^{(j)}$ and $L^{(j)}(0)$ from $q(z|\mathcal{D}_{\mathbb{C}}, A)$ and $q(L(0)|\mathcal{D}_{\mathbb{C}}, A)$ respectively.

Using the moment matching, we can construct a Gaussian posterior approximation for the predictive distribution as

$$p(X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A) \approx \mathcal{N}(\mathcal{Z}_1, \mathcal{Z}_2),$$

where,

$$\mathcal{Z}_1 = \mathbb{E}[X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A],$$

$$\mathcal{Z}_2 = \mathbb{E}[X^2_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A] - (\mathbb{E}[X_{l^{\mathbb{T}}}(t^{\mathbb{T}})|t^{\mathbb{T}}, l^{\mathbb{T}}, \mathcal{D}_{\mathbb{C}}, A])^2.$$

Appendix A.2 Proofs

In the NDP4ND, given $\tilde{z}$ and $L_{l_i^{\mathbb{T}}}(0)$, the evolved latent state $L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})$ can be seen as a deterministic function dominated by $t_i^{\mathbb{T}}$ and $l_i^{\mathbb{T}}$, involving the interaction between nodes. Thus, we have the following Lemma to assist in proving our proposition.

Lemma A1. The evolved latent state $L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})$ can be seen as a deterministic function $\mathcal{F}(t_i^{\mathbb{T}}, l_i^{\mathbb{T}})$ for a given fixed $\tilde{z}$ and $L_{l_i^{\mathbb{T}}}(0)$.

Proof. We have the fact that

$$L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}}) = L_{l_i^{\mathbb{T}}}(0) + \int_{t_0}^{t_i^{\mathbb{T}}} \left(\boldsymbol{S}(L_{l_i^{\mathbb{T}}}(t), \tilde{z}) + \sum_{j=1}^{n} A_{l_i^{\mathbb{T}}, l_j^{\mathbb{T}}} \boldsymbol{I}(L_{l_i^{\mathbb{T}}}(t), L_{l_j^{\mathbb{T}}}(t), \tilde{z}) \right) dt,$$

where, only $\tilde{z}$ (global control variable) and $L_{l_i^{\mathbb{T}}}(0)$ (initial hidden states) are uncertain, while the rest are deterministic. Therefore, when both $\tilde{z}$ and $L_{l_i^{\mathbb{T}}}(0)$ are fixed, the above equation can be regarded as a deterministic function controlled by $t_i^{\mathbb{T}}$ and $l_i^{\mathbb{T}}$, i.e., $\mathcal{F}(t_i^{\mathbb{T}}, l_i^{\mathbb{T}})$. □

Proposition A1. NDP4ND satisfies the exchangeability and consistency conditions.

Proof. First, we prove that NDP4ND satisfies the exchangeability condition. Following Lemma A1, for any given $\tilde{z}$ and $L_{l_i^{\mathbb{T}}}(0)$, any permutation π of $\{1, 2, ..., N_{\mathbb{T}}\}$ on $t^{\mathbb{T}}_{1:N_{\mathbb{T}}}$ and $l^{\mathbb{T}}_{1:N_{\mathbb{T}}}$ would automatically act on $\mathcal{F}_{1:N_{\mathbb{T}}}$, i.e.,

$$\mathcal{F}_{\pi(1:N_{\mathbb{T}})} = \mathcal{F}(t^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}, l^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}),$$

where $\pi(1:N_{\mathbb{T}}) = (\pi(1), \pi(2), ..., \pi(N_{\mathbb{T}}))$. And then the permutation π would act on the conditional predictive distribution, i.e.,

$$[\mu_{X_{l^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}}}(t^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}); \sigma^2_{X_{l^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}}}(t^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})})] = \boldsymbol{d}(t^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}, \mathcal{F}_{\pi(1:N_{\mathbb{T}})}, \tilde{z}).$$

Consequently, the permutation π would act on the predictive distribution, i.e., $p(X_{l^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}}(t^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})})|t^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}, l^{\mathbb{T}}_{\pi(1:N_{\mathbb{T}})}, \mathcal{D}_{\mathbb{C}}, A)$, because $\mathcal{Z}_1$ and $\mathcal{Z}_2$ in the distribution can also be seen as two deterministic functions dominated by $t_i^{\mathbb{T}}$ and $l_i^{\mathbb{T}}$. Therefore, the exchangeability condition is guaranteed.

Then, we prove that NDP4ND satisfies the consistency condition. Based on Lemma A1, we can write the joint distribution similar to NPs [3] or NDPs [4] as follows

$$\rho_{t^{\mathbb{T}}_{1:N_{\mathbb{T}}}, l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}})) = \int p(\mathcal{F}) \prod_{i=1}^{N_{\mathbb{T}}} p(X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})|\mathcal{F}(t_i^{\mathbb{T}}, l_i^{\mathbb{T}}))d\mathcal{F}.$$

Since the probability density function of any $X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})$ depends only on the corresponding pair $t_i^{\mathbb{T}}$ and $l_i^{\mathbb{T}}$, integrating out any subset of $X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}})$ gives the joint distribution of the remaining random variables in the sequence as

$$\begin{aligned}
&\int \rho_{t^{\mathbb{T}}_{1:N_{\mathbb{T}}}, l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(X_{l^{\mathbb{T}}_{1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{1:N_{\mathbb{T}}}))dX_{l^{\mathbb{T}}_{n+1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{n+1:N_{\mathbb{T}}}) \\
&= \int\int p(\mathcal{F}) \prod_{i=1}^{N_{\mathbb{T}}} p(X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})|\mathcal{F}(t_i^{\mathbb{T}}, l_i^{\mathbb{T}}))d\mathcal{F}dX_{l^{\mathbb{T}}_{n+1:N_{\mathbb{T}}}}(t^{\mathbb{T}}_{n+1:N_{\mathbb{T}}}) \\
&= \int p(\mathcal{F}) \prod_{i=1}^{n} p(X_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})|\mathcal{F}(t_i^{\mathbb{T}}, l_i^{\mathbb{T}}))d\mathcal{F} \\
&= \rho_{t^{\mathbb{T}}_{1:n}, l^{\mathbb{T}}_{1:n}}(X_{l^{\mathbb{T}}_{1:n}}(t^{\mathbb{T}}_{1:n})).
\end{aligned}$$

Therefore, consistency is also guaranteed. □

Appendix A.3 Computational complexity analysis

Our proposed model has two execution stages: training and predicting. The former is to learn parameters in model from various network ODE instances in the specific scenarios. The latter is to test on network ODE instances that have not been seen during training and predict future time series on the network based on sparse observations. Note that once our model is trained in a certain scenario, testing any unseen network ODE instances in that scenario does not require retraining. This is the core distinction between our model and other instance-oriented methods.

Now, we analyze the complexity of our model in dealing with unseen network emerging dynamics. As shown in Figure 2b-e in the main text, our model performs four key steps, and we analyze the complexity of each step separately.

1) Generate $p(z|\mathcal{D}_{\mathbb{C}}, A)$ based on observations by encoding temporal and topological information. The dominant part of the calculation in this step is mainly to use the GAT for encoding observations. The complexity of the GAT is $\mathcal{O}(K_{head} \times d_h \times d_h \times$

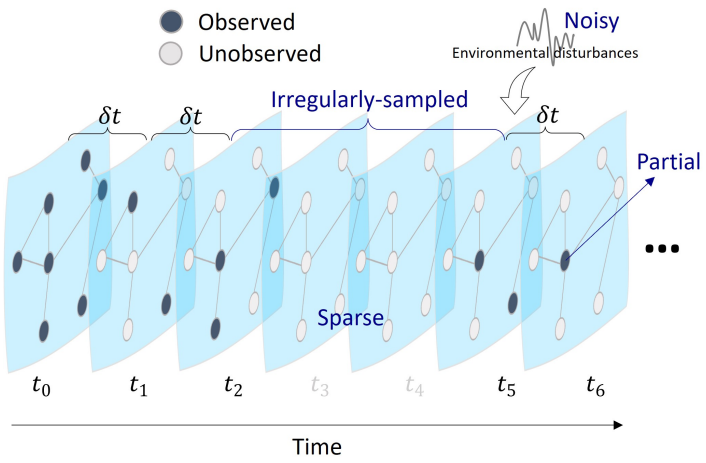

Figure B1 An illustration to show that the observations triggering predictions are sparse, irregularly-sampled, partial, and noisy.

$n) + \mathcal{O}(K_{head} \times d_h \times E)$, where, d_h is the hidden dimension, K_{head} is the number of heads, n and E are the numbers of nodes and edges in networked system, respectively. Thus, the main computational complexity in this step is $\mathcal{O}(K_{layer} \times K_{head} \times d_h \times d_h \times n \times N_{steps}) + \mathcal{O}(K_{layer} \times K_{head} \times d_h \times E \times N_{steps})$, where K_{layer} is the number of GAT layers and N_{steps} is the number of time steps in observations.

2) Generate $p(L(0)|\mathcal{D}_{\mathbb{C}}, A)$ based on observations. The dominant part of the calculation in this step is to encode initial states. We use a multi-layer perceptron to generate the distribution. Thus, the main computational complexity in this step is $\mathcal{O}(K_{layer} \times d_h \times d_h \times n)$, where K_{layer} is the number of hidden layers, d_h is the hidden dimension, and n is the numbers of nodes in networked system.

3) Calculate the differentiation of latent states $\dot{L}_l(t)$ on any node l at any time t. The dominant part of the calculation in this step is to compute the differentiation of latent states by neural networks S and I. Thus, the computational complexity for computing a differentiation on node l at time t is $\mathcal{O}(K_{layer} \times d_h \times d_h)$, where K_{layer} is the number of hidden layers and d_h is the hidden dimension.

4) Calculate latent states $L_l(t)$ and output the state distribution $p(X_l(t)|t, l, z, L(0), A)$ on any node l at any time t. The dominant part of the calculation in this step is to compute the latent states by solving an initial value problem and decode the latent states to system states in the original space. Thus, the computational complexity in this step for one node and one time step is $\mathcal{O}(K_{layer} \times d_h \times d_h \times P)$, where P is the order of the solver, which is related to the accuracy of the solution. If we need to predict T time steps and n nodes, then the main computational complexity in both step 3 and step 4 will reach $\mathcal{O}(K_{layer} \times d_h \times d_h \times P \times n \times T)$.

By integrating the complexity of the above parts, the overall computational complexity $\mathcal{C}$ is $\mathcal{O}(K_{layer} \times K_{head} \times d_h \times d_h \times n \times N_{steps}) + \mathcal{O}(K_{layer} \times K_{head} \times d_h \times E \times N_{steps}) + \mathcal{O}(K_{layer} \times d_h \times d_h \times P \times n \times T)$. Note that only n, E, N_{steps} and T will vary with the network system being processed, while the rest are hyper-parameters that remain unchanged once trained. That is to say, the complexity of the model is influenced by the system scale (including both the number of nodes and the number of edges), the number of predicted time steps, and the number of observed time steps. Actually, the complexity of our model in dealing with unseen network emerging dynamics increases linearly with the system scale.

Due to the fact that other network dynamics learning methods require both training and prediction for dealing with unseen network emerging dynamics, they typically have high computational complexity. The Figure 3i in the main text demonstrates that our model can be three orders of magnitude faster than existing methods.

Appendix B Experimental Details

In this section, we introduce specific experimental details, including the construction for training and testing sets, the detailed description of the network dynamics scenarios, baselines, metrics, and experimental setup.

Appendix B.1 Construction for training and testing sets

We assign randomly sampled values from the parameter to the tunable parameters in the ODE templates to obtain massive network ODE instances for each network dynamics scenario. Based on different ODE instances, topologies, and initial states, we randomly generate N_{Tr} training tasks and N_{Te} testing tasks per scenario. Note that the sampled ODE instances to produce testing tasks have not appeared in the training set. Therefore, for the model, the testing tasks to be handled come from the emerging dynamics. We irregularly and sparsely sample observations from each testing task to trigger predictions (Figure B1).

Appendix B.2 Network dynamics scenarios

We carried out extensive experiments on various complex network dynamics scenarios, including mutualistic interaction dynamics in ecosystems, second-order phototaxis dynamics, brain dynamics, compartment models in epidemiology, and empirical systems.

Appendix B.2.1 *Mutualistic interacting dynamics*

Mutualistic interacting dynamics among species in ecology [8] capture the abundance $X_i(t)$ of species i. The differential equation that characterizes abundance changes is as follows

$$\dot{X}_i(t) = b + X_i(t)(1 - \frac{X_i(t)}{k})(\frac{X_i(t)}{c} - 1) + \sum_j A_{i,j} \frac{X_i(t)X_j(t)}{d + eX_i(t) + hX_j(t)},$$

where b accounts for the rate of the incoming migration from neighboring ecosystems. The second term describes logistic growth with the system carrying capacity k and the Allee effect with negative growth threshold c. The third term describes mutualistic interactions, captured by a response function that saturates for large $X_i(t)$ or $X_j(t)$, indicating that j's positive contribution to $X_i(t)$ is bounded.

The detailed parameter setting for mutualistic interacting dynamics in epidemiology is shown in Table B1.

Table B1 Parameter setting for Mutualistic interacting dynamics

Network space $\mathcal{A}$	Parameter space Φ	# Nodes n	Initial state space $\mathcal{I}$
Grid Power law Random Small world Community	$b \in [0.05, 0.15]$, $c \in [0.5, 1.5]$, $d \in [4, 6]$, $e \in [0.8, 1.0]$, $h \in [0.05, 0.15]$, $k \in [4, 6]$.	$\mathcal{U}^{\dagger}[100, 200]$	$\mathcal{U}[0, 25]$
# Training tasks N_{Tr}	# Testing tasks N_{Te}	Min. sampling interval δt	Max. observed time T_o
1,500	100	1	50

Appendix B.2.2 *Second-order phototaxis dynamics*

Phototaxis dynamics is a second-order system simulating the dynamics of phototactic bacteria towards a fixed light source [9], which can be applied to movement of bacteria towards food sources [10] and emergency evacuation modeling [11]. The system performs bacteria's movements through the excitation level of bacteria to the light and their interactions with each other [10]. The system has five observed states: coordinate 1 ($X_{i,0}(t)$), coordinate 2 ($X_{i,1}(t)$), velocity 1 ($X_{i,2}(t)$), velocity 2 ($X_{i,3}(t)$) and excitation level ($X_{i,4}(t)$). The differential equations that characterize system behavior are as follows

$$\begin{aligned}
\dot{X}_{i,0:1}(t) =& X_{i,2:3}(t),\\
\dot{X}_{i,2:3}(t) =& \frac{\lambda_1}{n}\sum_{j=1}^{n} k_1(X_{j,0:1}(t), X_{i,0:1}(t))(X_{j,2:3}(t) - X_{i,2:3}(t))\\
&+ I_0(V - X_{i,2:3}(t))(1 - \varphi(X_{i,4}(t); \zeta_{cr})),\\
\dot{X}_{i,4}(t) =& \frac{\lambda_2}{n}\sum_{j=1}^{n} k_2(X_{j,0:1}(t), X_{i,0:1}(t))(X_{j,4}(t) - X_{i,4}(t)) + I_0\varphi(X_{i,4}(t); \zeta_{cp}),
\end{aligned}$$

where,

$$k_1(x_j, x_i) = k_2(x_j, x_i) = \frac{1}{(1 + |x_j - x_i|^2)^{\beta}},$$

$$\varphi(\zeta; \zeta_c) = \begin{cases} 1, & 0 \leqslant \zeta < \zeta_c, \\ 0.5(\cos(\frac{\pi}{\zeta_c}(\zeta - \zeta_c)) + 1), & \zeta_c \leqslant \zeta < 2\zeta_c, \\ 0, & 2\zeta_c \leqslant \zeta. \end{cases}$$

k_1 and k_2 are the alignment-based interaction kernels, φ is the smooth cutoff function, I_0 is the light intensity, V is the terminal velocity (light source at infinity), ζ_{cr} is the critical excitation level (when the light effect activates the bacteria), and ζ_{cp} is the excitation capacity.

The detailed parameter setting for the second-order phototaxis dynamics is shown in Table B2.

Table B2 Parameter setting for Second-order phototaxis dynamics

Network space $\mathcal{A}$	Parameter space Φ	# Nodes n	Initial state space $\mathcal{I}$
Complete graph	$\lambda_1 = \lambda_2 = 100$, $V = [60, 0]$, $\zeta_{cp} = 2\zeta_{cr}$, $I_0 \in [0.01, 1.0]$, $\beta \in [-0.4, -0.1]$, $\zeta_{cr} \in [0.1, 0.5]$.	$\mathcal{U}^{\dagger}[20, 50]$	$X_{i,0:3} \sim \mathcal{U}[0, 100]^4$ $X_{i,4} \sim \mathcal{U}[0, 1\text{e-}3]$
# Training tasks N_{Tr}	# Testing tasks N_{Te}	Min. sampling interval δt	Max. observed time T_o
300	20	0.01	0.5

Appendix B.2.3 *Brain dynamics*

We consider a kind of neuronal dynamics dynamics that brain activities are governed by the FitzHugh-Nagumo dynamics [12], which has a quasi-periodic characteristic. Topological structures with power-law distribution are used to simulate the interactions

among brain functional areas [13]. The dynamics capture the firing behaviors of neurons with two components, i.e., membrane potential ($X_{i,0}(t)$) and recovery variable ($X_{i,1}(t)$). The differential equations that characterize brain activities are as follows

$$\begin{aligned}\dot{X}_{i,0}(t) =& X_{i,0}(t) - X_{i,0}(t)^3 - X_{i,1}(t) - \epsilon \sum_{j=1}^{n} A_{i,j} \frac{X_{j,0}(t) - X_{i,0}(t)}{Deg_i^{in}},\\ \dot{X}_{i,1}(t) =& a + bX_{i,0}(t) + cX_{i,1}(t),\end{aligned}$$

where, Deg_i^{in} is the indegree of node i.

The detailed parameter setting for brain dynamics is shown in Table B3.

Table B3 Parameter setting for Brain dynamics

Network space $\mathcal{A}$	Parameter space Φ	# Nodes n	Initial state space $\mathcal{I}$
Power law	$a \in [0.2, 0.3]$, $b \in [0.4, 0.6]$, $c \in [-0.06, -0.02]$, $\epsilon \in [0.8, 1.2]$.	$\mathcal{U}^{\dagger}[100, 200]$	$\mathcal{U}[-1, 1]^2$
# Training tasks N_{Tr}	# Testing tasks N_{Te}	Min. sampling interval δt	Max. observed time T_o
300	20	0.01	0.5

Appendix B.2.4 *Compartment models in epidemiology*

Compartment models [14, 15] are often used to simulate the spread of epidemics on networks. Herein, we conduct experiments on three commonly used compartment models: susceptible-infectious-susceptible (SIS), susceptible-infectious-recovered (SIR), and susceptible-exposed-infectious-susceptible (SEIS).

SIS: The susceptible-infectious-susceptible (SIS) model has two states, i.e., susceptible and infectious. This model assumes that individual states not only transition from susceptible to infectious with an infection rate b but also reverse with a recovery rate r. The differential equations that characterize the model are as follows

$$\begin{aligned}S : \dot{X}_{i,0}(t) &= -b \times X_{i,0}(t) \times \sum_j A_{i,j} X_{j,1}(t) + r \times X_{i,1}(t),\\ I : \dot{X}_{i,1}(t) &= b \times X_{i,0}(t) \times \sum_j A_{i,j} X_{j,1}(t) - r \times X_{i,1}(t).\end{aligned}$$

SIR: The susceptible-infectious-recovered (SIR) model has three states, i.e., susceptible, infectious, and recovered. The recovered state refers to an individual who has recovered from an epidemic. It assumes that 1) individual states transition from susceptible to infectious with an infection rate b; 2) individual states transition from infectious to recovered with a recovery rate r. In this model, the recovered individual has immunity, so he cannot be infected again. The differential equations that characterize the model are as follows

$$\begin{aligned}S : \dot{X}_{i,0}(t) &= -b \times X_{i,0}(t) \times \sum_j A_{i,j} X_{j,1}(t),\\ I : \dot{X}_{i,1}(t) &= b \times X_{i,0}(t) \times \sum_j A_{i,j} X_{j,1}(t) - r \times X_{i,1}(t),\\ R : \dot{X}_{i,2}(t) &= r \times X_{i,1}(t).\end{aligned}$$

SEIS: The susceptible-exposed-infectious-susceptible (SEIS) model has three states: susceptible, exposed, and infectious. The exposed state refers to an individual who has come into contact with infected individuals but is currently not contagious. The model assumes that 1) individual states transition from susceptible to exposed with an infection rate b; 2) individual states transition from exposed to infectious with an attack rate c; 3) individual states transition from infectious to susceptible with a recovery rate r. The differential equations that characterize the model are as follows

$$\begin{aligned}S : \dot{X}_{i,0}(t) &= -b \times X_{i,0}(t) \times \sum_j A_{i,j} X_{j,1}(t) + r \times X_{i,1}(t),\\ E : \dot{X}_{i,1}(t) &= b \times X_{i,0}(t) \times \sum_j A_{i,j} X_{j,1}(t) - c \times X_{i,1}(t),\\ I : \dot{X}_{i,2}(t) &= c \times X_{i,1}(t) - r \times X_{i,2}(t).\end{aligned}$$

The detailed parameter settings for compartment models in epidemiology are shown in Table B4.

Appendix B.2.5 *Empirical systems*

For the real-world systems, we collect daily global spreading data on three real diseases, including H1N1 [16], SARS [16], and COVID-19 [17], and use the worldwide airline network retrieved from OpenFights as a directed and weighted empirical topology. Only early data before government intervention, i.e., the first 50 days, is considered here to keep the spread features of the disease itself. We divide H1N1 data into 1,000 standardized training tasks according to various time intervals, where the time interval is sampled from $\{2, 3, ..., 21\}$ and the start time is sampled from $\{0, 1, ..., 49\}$. Then, we test methods on all three diseases. As a result, testing on H1N1 can be seen as predicting re-emerging infectious diseases, as the method trained on the same data but testing on SARS and COVID-19 as predicting emerging infectious diseases.

Table B4 Parameter setting for Compartment models in epidemiology (SIS, SIR, and SEIS)

Network space $\mathcal{A}$	Parameter space Φ	# Nodes n	Initial state space $\mathcal{I}$
Power law	For SIS and SIR: $b \in [0.02, 0.2]$, $r \in [0.1, 0.4]$. For SEIS: $b \in [0.3, 2.0]$, $r \in [0.1, 0.4]$, $c \in [0.05, 0.1]$.	$\mathcal{U}^{\dagger}[100, 200]$	For SIS: $S : X_{i,0} = 1 - X_{i,1}$, $I : X_{i,1} \sim \mathcal{U}$[1e-6, 1e-3] For SIR: $S : X_{i,0} = 1 - X_{i,1} - X_{i,2}$, $I : X_{i,1} \sim \mathcal{U}$[1e-6, 1e-3], $R : X_{i,2} = 0$ For SEIS: $S : X_{i,0} = 1 - X_{i,1} - X_{i,2}$, $E : X_{i,1} = 0$, $I : X_{i,2} \sim \mathcal{U}$[1e-6, 1e-3]
# Training tasks N_{Tr}	# Testing tasks N_{Te}	Min. sampling interval δt	Max. observed time T_o
1,000 per model	100 per model	1	50

Appendix B.2.6 *Settings for sparse and noisy observations*

To validate our method's tolerance for the observed ratio, we sample observations with various observed ratios from time interval $[0, T/2]$, where, T is the maximum predictive time step, and then test our method on the following observed ratios: 2.4%, 2.8%,3.2%, 3.6%, 4.0%, 6.0%, 8.0%, 10%, and 20%. We sample 20 new network ODE instances per observed ratio and report the predictive errors of our method over the entire time interval $[0, T]$. We also report the predictive errors of the NDCN and DNND with observed ratio of 60% for comparison. The results are reported in Figure C2(a).

To test the robustness of our method for the noise, we first sample 20 new network ODE instances, and then add white Gaussian disturbances to observations, i.e., $\mathcal{N}(0, \xi^2)$, resulting in various noise levels, i.e., $\xi \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. We report the predictive errors of the NDCN, DNND and our method over the entire time interval $[0, T]$ for all sampled network ODE instances under various noise levels in Figure C2(b).

Appendix B.3 Baselines

The methods we compared in the experiments are listed below.

LG-ODE [18]: The first work is to learn continuous-time system dynamics from irregularly-sampled partial observations, which is to use a carefully-designed spatio-temporal transformer as the encoder and use a graph neural network to model the network dynamic equation.

NDCN [19]: A representative neural dynamics method, explicitly proposed for automatically modeling network dynamics. Its main idea is to use a graph neural network to model the network dynamic equation in the hidden space mapped by the neural encoder, and can learn continuous-time dynamics on complex networks with irregularly-sampled observations.

DNND [20]: A state-of-the-art neural dynamics method for automatically modeling network dynamics. It alleviates the limitations of the NDCN in long-term prediction by abandoning the encoder and decoder, directly models network differential equations in the state space, and simplifies the design of neural networks using first principles, achieving state-of-the-art results.

NP [3]: Stochastic processes parameterized by neural networks.

NDP [4]: A class of stochastic processes, generalized NPs defined over time by combining latent neural ODEs with NPs.

NDP4ND w/o ode: A variant of our NDP4ND, removing ODE flow in the architecture, i.e., deleting $L_{l_i^{\mathbb{T}}}(t_i^{\mathbb{T}})$ from the input of neural network $\boldsymbol{d}$.

NDP4ND w/o z: A variant of our NDP4ND, removing z from the input of neural network $\boldsymbol{d}$ in the architecture.

We note that NPs and NDPs are not specifically designed for network dynamics learning. Using them as comparison methods in experiments is to verify the necessity of introducing interactions of time dynamics on networks.

Appendix B.4 Metrics

The following metrics are used for performance comparison:

Mean Absolute Error (MAE): MAE quantifies the mean absolute error between the predictions and the true values over all nodes and at all times, which can be calculated as

$$\frac{1}{nTD}\sum_{i=1}^{n}\sum_{t=1}^{T}\sum_{d=1}^{D}|X_i(t)[d] - \hat{X}_i(t)[d]|,$$

where, n, T, and D are the number of nodes in the system, the maximum prediction time, and the dimension of observable states, respectively. $\hat{X}_i(t)[d]$ is the d-th dimension predictive state on node i at time t and $X_i(t)[d]$ is the ground truth.

Dynamic Time Warping (DTW): DTW quantifies the similarity between predictive system dynamic behavior and ground truth, which can be calculated as

$$\frac{1}{nD}\sum_{i=1}^{n}\sum_{d=1}^{D} dtw(X_i(.)[d] - \hat{X}_i(.)[d]),$$

where, n and D are the number of nodes in the system and the dimension of observable states, respectively. $dtw(.)$ is a FastDTW method [21] that is an approximation of dynamic time warping with linear time and space complexity.

Observed ratio: We define the observed ratio as $\frac{N_{\mathbb{C}}}{((t_o - t_s)/\delta t + 1) \times \#nodes} \times 100\%$ to quantify the sampling density of observations, where $N_{\mathbb{C}}$ is the number of observations, δt is the minimum sampling interval, t_s is the start time, and t_o is maximum time of all observations.

Appendix B.5 Experimental setup

In the training phase, we use the Adam algorithm [22] to optimize parameters in the model. Specifically, the initial learning rate is set to 5e-3, followed by an exponential decay with a decay coefficient of 10. The batch size and epochs are set to 8 and 200, respectively. All experiments were conducted on the hardware environment with the same computational resources, including Intel(R) Xeon(R) Silver 4310 CPU@2.10GHz $\times$ 48 and NVIDIA GeForce RTX 3090 $\times$ 3.

Appendix C Addtional results

Appendix C.1 Tables

Table C1 The interpolation and extrapolation results across all testing network ODEs on *Second-order Phototaxis* dynamics in terms of quantitative predictive error and similarity. The reported values are the *mean* (*standard deviation*) of the Mean Absolute Error (MAE) and Dynamic Time Warping (DTW) between prediction and ground truth. The best results are bolded.

Methods	Metrics	Interpolation ($T_o \leqslant 0.5$)	Extrapolation ($T_o > 0.5$)
NDCN	MAE	5.78E+00 (1.67E+00)	1.31E+01 (9.77E+00)
	DTW	2.49E+02 (8.36E+01)	6.31E+02 (4.97E+02)
DNND	MAE	9.92E-01 (2.65E-01)	2.85E+00 (1.17E+00)
	DTW	1.74E+01 (1.22E+01)	1.18E+02 (5.53E+01)
NDP4ND(Ours)	MAE	**6.00E-01 (1.18E-01)**	**9.09E-01 (4.87E-01)**
	DTW	**1.48E+01 (2.89E+00)**	**2.87E+01 (1.61E+01)**

Table C2 The interpolation and extrapolation results across all testing network ODEs on *Brain* dynamics in terms of quantitative predictive error and similarity. The reported values are the *mean* (*standard deviation*) of the Mean Absolute Error (MAE) and Dynamic Time Warping (DTW) between prediction and ground truth. The best results are bolded.

Methods	Metrics	Interpolation ($T_o \leqslant 0.5$)	Extrapolation ($T_o > 0.5$)
NDCN	MAE	1.79E+00 (8.85E-01)	2.65E+01 (2.32E+01)
	DTW	7.30E+01 (4.14E+01)	1.32E+03 (1.16E+03)
DNND	MAE	1.03E+00 (1.21E-01)	1.48E+00 (2.35E-01)
	DTW	4.01E+01 (1.35E+01)	5.79E+01 (6.98E+00)
NDP4ND(Ours)	MAE	**2.91E-01 (3.64E-02)**	**6.06E-01 (1.02E-01)**
	DTW	**6.66E+00 (1.08E+00)**	**1.46E+01 (1.54E+00)**

Table C3 The interpolation and extrapolation results across all testing network ODEs on *Compartment Models* in epidemiology in terms of quantitative predictive error and similarity. The reported values are the *mean* (*standard deviation*) of the Mean Absolute Error (MAE) and Dynamic Time Warping (DTW) between prediction and ground truth. The best results are bolded.

Dynamics	Methods	Metrics	Interpolation ($T_o \leqslant 50$)	Extrapolation ($T_o > 50$)
SIS	NDCN	MAE	**6.32E-02 (3.94E-02)**	4.95E-01 (6.48E-01)
		DTW	**1.81E+00 (1.40E+00)**	2.48E+01 (3.24E+01)
	DNND	MAE	1.27E-01 (1.79E-01)	1.13E+01 (4.88E+01)
		DTW	4.03E+00 (7.96E+00)	5.64E+02 (2.44E+03)
	NDP4ND(Ours)	MAE	1.04E-01 (7.91E-02)	**4.45E-02 (4.34E-02)**
		DTW	1.93E+00 (1.88E+00)	**2.20E+00 (2.19E+00)**
SIR	NDCN	MAE	**1.27E-01 (1.79E-01)**	3.32E-01 (4.26E-01)
		DTW	5.28E+00 (9.03E+00)	1.66E+01 (2.13E+01)
	DNND	MAE	5.62E+00 (2.41E+01)	1.64E+04 (7.14E+04)
		DTW	2.81E+02 (1.21E+03)	8.18E+05 (3.57E+06)
	NDP4ND(Ours)	MAE	1.51E-01 (8.69E-02)	**5.41E-02 (8.16E-02)**
		DTW	**3.39E+00 (3.69E+00)**	**2.70E+00 (4.08E+00)**
SEIS	NDCN	MAE	**1.02E-01 (1.23E-01)**	9.42E-01 (1.25E+00)
		DTW	3.49E+00 (6.18E+00)	4.71E+01 (6.24E+01)
	DNND	MAE	2.12E-01 (2.71E-01)	3.54E-01 (6.51E-01)
		DTW	7.21E+00 (1.37E+01)	1.77E+01 (3.26E+01)
	NDP4ND(Ours)	MAE	1.03E-01 (7.32E-02)	**3.14E-02 (2.12E-02)**
		DTW	**1.27E+00 (8.87E-01)**	**1.57E+00 (1.06E+00)**

Appendix C.2 Figures

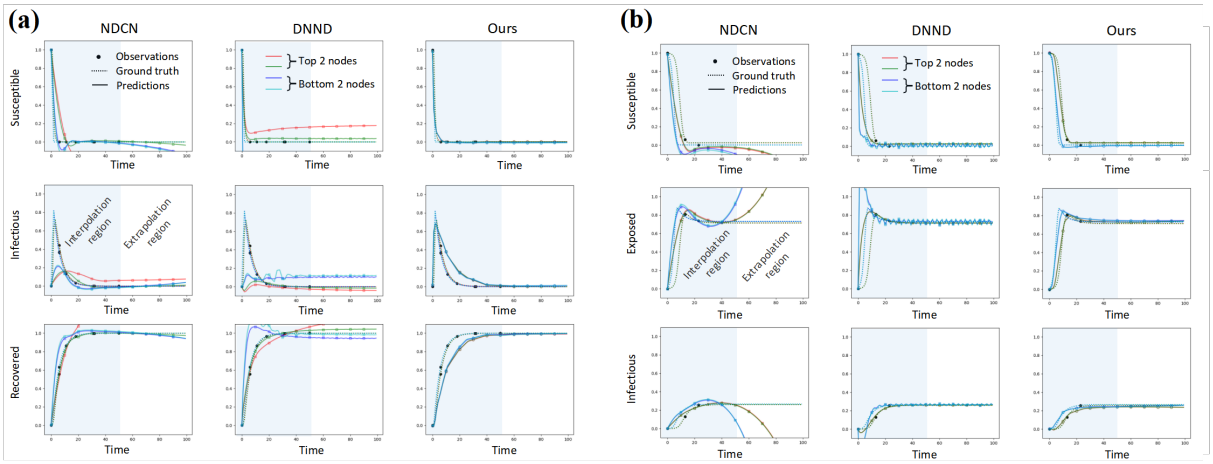

Figure C1 The interpolation and extrapolation results on *susceptible-infectious-recovered (SIR)* and *susceptible-exposed-infectious-susceptible (SEIS)*. **(a)** The testing results of the NDCN, DNND, and our NDP4ND on the SIR. The ratio of observations in this testing network ODE is 6.89%. **(b)** The testing results of the NDCN, DNND, and our NDP4ND on the SEIS. The ratio of observations in this testing network ODE is 4.08%. The number of nodes and the maximum value of all observed times (T_o) are 200 and 50, respectively. Overall, our NDP4ND can learn the most effective network dynamics from irregularly-sampled, partial, and sparse observations.

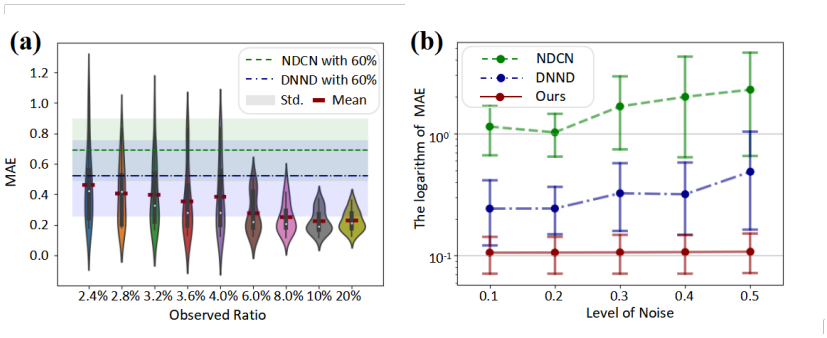

Figure C2 **(a)** The testing results on mutualistic interaction dynamics at various observed ratios. As the observation density increases, the predictive error of the NDP4ND appears a decreasing trend, where the prediction error occurs a cliff like decline when the observed ratio is around 6%, reaching around 0.3. By contrast, the prediction errors of the LG-ODE, NDCN, and DNND at an observed ratio of 60% are 3.99±3.59, 0.70±0.20, and 0.52±0.24, respectively. **(b)** The testing results on mutualistic interaction dynamics at various noise levels. As the noise level increases, the predictive errors shows an increasing trend, demonstrating that the introduction of noise indeed weakens predictions. Benefiting from uncertainty modeling, our NDP4ND is relatively stable and has significantly outperformed others.

• • • • • • • • • • • • • • •

References

1 Barzel B, Barabási A-L. Universality in network dynamics. Nature Physics, 2013, 9: 673–681
2 Chen R T Q, Rubanova Y, Bettencourt J, et al. Neural ordinary differential equations. In: Proceedings of Advances in Neural Information Processing Systems, 2018, 31
3 Garnelo M, chwarz J, Rosenbaum D, et al. Neural processes. In: Proceedings of ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018
4 Norcliffe A, Bodnar C, Day B, et al. Neural ODE processes. In: Proceedings of International Conference on Learning Representations, 2021.
5 Veličković P, Cucurull G, Casanova A, et al. Graph Attention Networks. In: Proceedings of International Conference on Learning Representations, 2018.
6 Cui J, Yang B. Survey on Bayesian Optimization Methodology and Applications. Journal of Software, 2018, 29(10):3068-3090
7 Garnelo M, Rosenbaum D, Maddison C, et al. Conditional neural processes. In: Proceedings of International Conference on Machine Learning, 2018, 80: 1704–1713
8 Gao J, Barzel B, Barabási A-L. Universal resilience patterns in complex networks. Nature, 2016, 530: 307–312
9 Ha S-Y, Levy D. Particle, kinetic and fluid models for phototaxis. Discrete and Continuous Dynamical Systems - B, 2009, 12: 77–108
10 Lu F, Zhong M, Tang S, et al. Nonparametric inference of interaction laws in systems of agents from trajectory data. Proceedings of the National Academy of Sciences, 2019, 116: 14424–14433
11 Lin J, Lucas T A. A particle swarm optimization model of emergency airplane evacuations with emotion. Networks and Heterogeneous Media, 2015, 10: 631–646
12 Gerster M, Berner R, Sawicki J, et al. Fitzhugh-nagumo oscillators on complex networks mimic epileptic-seizure-related synchronization phenomena. Chaos,2020, 30: 123130
13 Gao T, Yan G. Autonomous inference of complex network dynamics from incomplete and noisy data. Nature Computational Science, 2022, 2: 160–168.
14 Dimitrov N B, Meyers L A. Mathematical Approaches to Infectious Disease Prediction and Control, INFORMS, 2010, Chapter 1, 1–25
15 Youssef M, Scoglio C. An individual-based approach to sir epidemics in contact networks. Journal of Theoretical Biology, 2011, 283: 136–144
16 Nunes L. A brief comparative study of epidemics. Online website, 2020, https://www.kaggle.com/code/lnunes/a-brief-comparative-study-of-epidemics
17 Dong E, Du H, Gardner L. An interactive web-based dashboard to track covid-19 in real time. The Lancet Infectious Diseases, 2020, 20: 533–534
18 Huang Z, Sun Y, Wang W. Learning continuous system dynamics from irregularly-sampled partial observations. In: Proceedings of Advances in Neural Information Processing Systems,2020, 33: 16177–16187
19 Zang C, Wang F. Neural dynamics on complex networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,2020, 892–902
20 Liu B, Luo W, Li G, et al. Do we need an encoder-decoder to model dynamical systems on networks? In: Proceedings of International Joint Conference on Artificial Intelligence, 2023, 2178–2186
21 Salvador S, Chan P K. Toward Accurate Dynamic Time Warping in Linear Time and Space. Intelligent Data Analysis, 2004, 11: 561–580
22 Kingma D P, Ba J. Adam: A Method for Stochastic Optimization. In: Proceedings of International Conference on Learning Representations, 2015