NeSyF: Neural-Symbolic Feedback for Joint Network Dynamics Prediction and Governing Equation Discovery

Anonymous authors
Paper under double-blind review

Abstract

Predicting network dynamics while recovering their governing equations remains challenging because existing neural-symbolic approaches often use recovered equations as final outputs or post-hoc interpretations, leaving the interaction between prediction and equation discovery largely one-way. We propose a Neural-Symbolic Feedback(NeSyF) framework that couples network dynamics prediction and governing equation discovery. The framework decomposes network dynamics into self-dynamics and interaction dynamics, learns them with separate neural networks, and applies symbolic regression to recover explicit equations. These equations are then used as derivative-consistency supervision to refine the learned dynamics. Experiments across four network topologies and four nonlinear dynamical systems demonstrate accurate long-horizon prediction and explicit equation recovery, while experiments on two real-world epidemic datasets further demonstrate accurate state prediction and yield compact symbolic models of empirical dynamics. A rollout-error analysis provides a theoretical rationale for derivative-consistency feedback, while ablation studies demonstrate its contribution to long-horizon predictive performance.

1 Introduction

Complex networked systems arise in diverse settings, including epidemic spreading, gene regulation, neural activity, and collective behavior, where system evolution is often governed by intrinsic node dynamics together with interactions structured by network topology (Boccaletti et al., 2006; Barzel & Barabási, 2013). Learning these dynamics from data is important for both predicting future system evolution and uncovering the mechanisms underlying observed behaviors. With the increasing availability of network topology and node-state trajectories, data-driven learning of complex network dynamics has emerged as an important problem in AI for scientific discovery (Wang et al., 2023). However, the governing mechanisms are often unobserved and must be inferred from finite observations. A central challenge is therefore to achieve accurate prediction while recovering explicit governing equations from the same observations.

A large body of work has explored data-driven prediction of network dynamics. Early approaches approximated state evolution using linear operators in reduced-order representations (Schmid, 2010), while subsequent deep models learned nonlinear temporal dependencies and topology-induced interactions (Li et al., 2018). Graph neural networks and continuous-time models have further advanced nonlinear dynamics modeling (Zang & Wang, 2020), while recent methods improve long-horizon prediction through structure-aware representations (Li et al., 2024) and multiscale representations (Li et al., 2025). In these prediction-oriented neural models, however, the learned dynamics remain largely implicit in model parameters, limiting direct access to explicit governing equations and mechanistic interpretation. In parallel, governing equation discovery seeks explicit dynamical laws from observations (Sahimi, 2024), with recent neural-symbolic methods decomposing network dynamics into self-dynamics and interaction dynamics to make symbolic inference more tractable (Gao et al., 2024; Hu et al., 2025). However, recovered equations are generally the endpoint of the pipeline rather than being fed back to refine the neural dynamics model.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This motivates a natural question: Can recovered dynamical equations serve not only as interpretable outputs, but also as feedback supervision for improving network dynamics learning? We propose NeSyF, a neural-symbolic feedback framework that couples network dynamics prediction and governing equation discovery. Our contributions are as follows:

- We introduce NeSyF, a neural-symbolic feedback framework in which recovered dynamical equations are fed back to refine the neural predictor, enabling bidirectional interaction between prediction and equation discovery.
- We design equation-guided derivative-consistency feedback by aligning neural derivatives with those computed from the recovered dynamics, and provide a conditional Euler error-propagation analysis relating neural–symbolic vector-field discrepancy to long-horizon rollout error.
- We introduce a progressive feedback strategy based on warm-start initialization, loss-scale calibration, and linear warm-up to improve the stability of symbolic feedback training.
- Extensive experiments demonstrate accurate long-horizon prediction and explicit equation recovery on synthetic network dynamics, while real-world epidemic experiments show accurate state prediction and yield compact, interpretable symbolic models. Ablation studies further verify the contribution of symbolic feedback to predictive performance.

2 Related Work

Network dynamics prediction. Data-driven approaches to network dynamics prediction learn temporal state evolution from observed trajectories, with early graph-based models jointly capturing spatial and temporal dependencies in graph-structured data (Li et al., 2018; Yu et al., 2018). Neural differential equations further enabled continuous-time modeling, with models such as NDCN combining graph neural networks with ordinary differential equations to learn dynamics on complex networks (Chen et al., 2018; Zang & Wang, 2020). Subsequent work has emphasized the importance of reliable long-term dynamics: DNND shows that models can fit observations well yet exhibit incorrect long-term behavior, motivating direct modeling of vector-field components in the original state space (Liu et al., 2023). Related work has also studied generalization across unseen graphs, out-of-distribution states, and varying environments (Huang et al., 2023; Vasiliauskaite & Antulov-Fantulin, 2024). More recent methods address sparse or imperfect observations, as well as long-horizon prediction, through stochastic, structure-aware, or multiscale representations (Cui et al., 2024; Li et al., 2024; 2025). Despite these advances, learned dynamics in prediction-oriented models remain largely implicit in neural parameters, limiting direct access to explicit governing equations and mechanistic interpretation.

Governing equation discovery. Governing equation discovery seeks to recover compact, explicit dynamical laws from observations. Classical approaches such as SINDy identify relevant terms through sparse regression over predefined function libraries (Brunton et al., 2016), while AI Feynman and PySR extend symbolic regression through physics-inspired decomposition and evolutionary search, respectively (Udrescu & Tegmark, 2020; Cranmer, 2023). Beyond these classical methods, learning-based symbolic regression explores equation spaces using deep generative models and transformers (Holt et al., 2023; d'Ascoli et al., 2024). For high-dimensional networked systems, however, directly searching for complete equations in the original variable space can be computationally challenging. Recent approaches therefore exploit structural priors, neural representations, and dimensionality reduction. Graph-based neural-symbolic methods apply symbolic regression to learned internal functions (Cranmer et al., 2020), while other methods decompose network dynamics into lower-dimensional dynamical components before symbolic discovery (Gao & Yan, 2022; Gao et al., 2024; Hu et al., 2025). More recent work further reduces high-dimensional symbolic search to lower-dimensional problems and uses pretrained neural models to guide formula discovery (Yu et al., 2026). These advances improve equation discovery, but related neural-symbolic pipelines generally use recovered equations as final outputs rather than feeding them back to refine the neural dynamics model, leaving the information flow predominantly one-way.

3 Problem Formulation

Consider a networked dynamical system with known topology $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, where $|\mathcal{V}| = N$, $\boldsymbol{A} = [A_{ij}] \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix, A_{ij} denotes the interaction strength from

node j to node i, and $\mathcal{N}_i$ denotes the set of in-neighbors of node i. Let $\boldsymbol{x}_i(t) \in \mathbb{R}^d$ denote the state of node i at time t, and define the network state as $\boldsymbol{X}(t) = [\boldsymbol{x}_1(t)^\top; \ldots; \boldsymbol{x}_N(t)^\top] \in \mathbb{R}^{N\times d}$. At discrete sampling times $t_k = k\Delta t$, we observe a finite state sequence $\{\boldsymbol{X}_k\}_{k=0}^{T-1}$, where $\boldsymbol{X}_k = \boldsymbol{X}(t_k)$.

We consider network dynamics of the form

$$\dot{\boldsymbol{x}}_i(t) = f(\boldsymbol{x}_i(t)) + \sum_{j\in\mathcal{N}_i} A_{ij} g(\boldsymbol{x}_i(t), \boldsymbol{x}_j(t)), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ describes the self-dynamics of an individual node, and $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ describes the pairwise interaction dynamics between connected nodes.

Given the network topology $\boldsymbol{A}$ and a finite sequence of state observations $\{\boldsymbol{X}_k\}_{k=0}^{T-1}$, our goal is to learn the network dynamics for future-state prediction while recovering explicit governing equations for the self-dynamics f and interaction dynamics g. We further use the recovered equations to constrain and refine the learned dynamics, thereby coupling network dynamics prediction and governing equation discovery.

4 METHOD

The overall NeSyF framework is shown in Fig. 1 and consists of three sequential stages. Stage I learns the self-dynamics and interaction dynamics in Eq. (1) using separate neural networks, yielding the base neural model M_0. Stage II constructs symbolic regression samples from the self- and interaction-dynamics components learned by M_0 and recovers explicit equations for the self-dynamics and interaction dynamics. Stage III uses the recovered equations to provide derivative-consistency supervision for refining the neural dynamics model, producing the final model M_1, which serves as the final predictor of NeSyF. Pseudocode for the complete training procedure is provided in Appendix A.

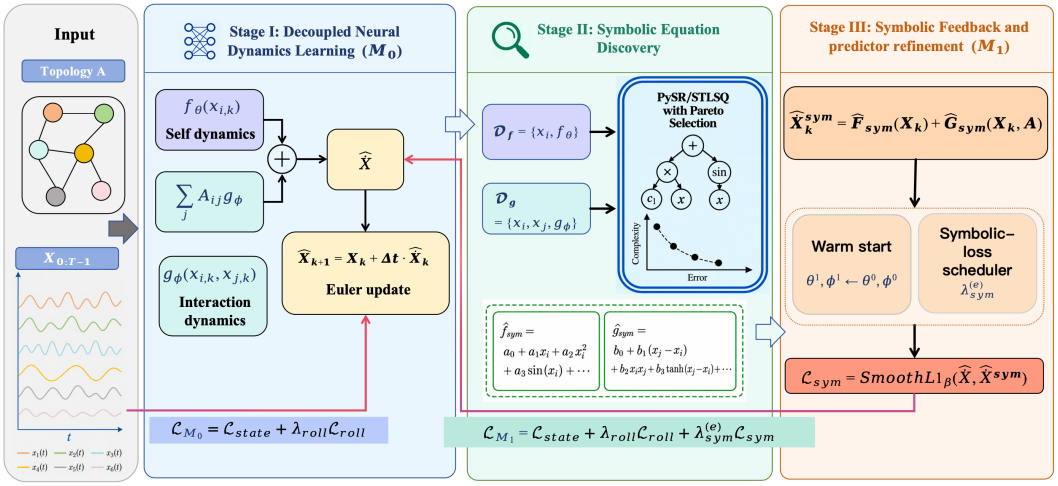


Figure 1: Overview of the NeSyF neural-symbolic feedback framework.

4.1 STAGE I: DECOUPLED NEURAL DYNAMICS LEARNING

In this stage, separate neural networks learn the self-dynamics function f and interaction dynamics function g in Eq. (1). Let $\boldsymbol{x}_{i,k} = \boldsymbol{x}_i(t_k)$. The predicted derivative of node i at time index k is

$$\hat{\dot{\boldsymbol{x}}}_{i,k} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i,k}) + \sum_{j\in\mathcal{N}_i} A_{ij} g_{\boldsymbol{\phi}}(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}), \tag{2}$$

where $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^d$ and $g_{\boldsymbol{\phi}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ denote neural parameterizations of the self-dynamics and interaction dynamics functions, respectively.

To maintain short-horizon state prediction accuracy while mitigating error accumulation during multi-step rollouts, Stage I jointly uses one-step state supervision and multi-step rollout supervision. The training objective is

$$\mathcal{L}_{\mathrm{M}_0} = \mathcal{L}_{\mathrm{state}} + \lambda_{\mathrm{roll}} \mathcal{L}_{\mathrm{roll}}, \tag{3}$$

where $\mathcal{L}_{\mathrm{state}}$ and $\mathcal{L}_{\mathrm{roll}}$ denote the one-step state loss and multi-step rollout loss, respectively, and $\lambda_{\mathrm{roll}} \geq 0$ is the rollout loss weight. Specifically,

$$\begin{aligned} \mathcal{L}_{\mathrm{state}} &= \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \mathrm{SL1}_\beta\left(\widehat{\boldsymbol{X}}_{k+1|k}, \boldsymbol{X}_{k+1}\right), \\ \mathcal{L}_{\mathrm{roll}} &= \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \sum_{h=2}^{H} \gamma^{h-1} \mathrm{SL1}_\beta\left(\widehat{\boldsymbol{X}}_{k+h|k}, \boldsymbol{X}_{k+h}\right). \end{aligned} \tag{4}$$

Here, $\mathcal{T}$ is the set of training starting-time indices, H is the rollout horizon during training, and $\gamma \in (0, 1]$ is the temporal decay factor across prediction horizons. $\widehat{\boldsymbol{X}}_{k+h|k}$ denotes the h-step rollout prediction starting from time index k, with $\widehat{\boldsymbol{X}}_{k|k} = \boldsymbol{X}_k$. The rollout states are obtained using explicit Euler integration:

$$\widehat{\boldsymbol{X}}_{k+h|k} = \widehat{\boldsymbol{X}}_{k+h-1|k} + \Delta t\, \widehat{\dot{\boldsymbol{X}}}_{k+h-1|k}, \qquad h = 1, \ldots, H. \tag{5}$$

Both supervision terms use the Smooth L1 loss, defined for arbitrary $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{N \times d}$ as

$$\mathrm{SL1}_\beta(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{Nd} \sum_{i=1}^{N} \sum_{q=1}^{d} \rho_\beta(U_{iq} - V_{iq}), \qquad \rho_\beta(r) = \begin{cases} \dfrac{r^2}{2\beta}, & |r| < \beta, \\ |r| - \dfrac{\beta}{2}, & |r| \geq \beta, \end{cases} \tag{6}$$

where $\beta = 1$ is the transition parameter of the Smooth L1 loss. Optimizing Eq. (3) yields the Stage-I base model, denoted M_0.

4.2 Stage II: Symbolic Equation Discovery

After fixing the Stage-I base model M_0, we recover explicit expressions for the self-dynamics and interaction dynamics from $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\phi}}$, respectively. The self-dynamics branch takes the node state $\boldsymbol{x}_{i,k}$ as input, while the interaction dynamics branch takes a pair of neighboring node states $(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k})$ as input. The corresponding neural outputs are used as symbolic regression targets. The two symbolic regression sample sets are

$$\mathcal{D}_f = \left\{\left(\boldsymbol{x}_{i,k}, f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i,k})\right)\right\}, \qquad \mathcal{D}_g = \left\{\left((\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}), g_{\boldsymbol{\phi}}(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k})\right) : j \in \mathcal{N}_i\right\}.$$

Here, $\mathcal{D}_f$ and $\mathcal{D}_g$ are used to recover the self-dynamics and interaction dynamics equations, respectively.

To exploit the complementarity of different symbolic search strategies, we use STLSQ (Brunton et al., 2016) and PySR (Cranmer, 2023) to generate candidate expressions of varying complexity. STLSQ performs sparse regression over a predefined library of candidate functions and uses sequential thresholding to obtain compact analytical expressions, whereas PySR uses symbolic search to explore more flexible nonlinear expression structures. Both methods generate candidate equations for the self-dynamics and interaction dynamics, and their union forms the candidate pool for subsequent equation selection.

For each candidate expression, we compute its validation regression error E_{reg} and expression complexity C and construct a Pareto front to balance fitting accuracy and expression simplicity. Among the Pareto-optimal candidates, we retain near-optimal expressions whose validation error is no greater than $(1 + \delta)$ times that of the best candidate and then select the least complex equation. We set $\delta = 0.05$, yielding the final self-dynamics equation $\widehat{f}_{\mathrm{sym}}$ and interaction dynamics equation $\widehat{g}_{\mathrm{sym}}$. Details of symbolic regression candidate generation and selection are provided in Appendix C.

4.3 Stage III: Symbolic Feedback and Predictor Refinement

Stage III initializes from the Stage-I base model M_0, keeps the dynamical equations recovered in Stage II fixed, and uses them to further refine the network dynamics prediction model. Specifically, substituting the recovered self-dynamics equation $\widehat{f}_{\mathrm{sym}}$ and interaction dynamics equation $\widehat{g}_{\mathrm{sym}}$ into Eq. (1) gives the symbolic derivative estimate of node i at time t_k:

$$\widehat{\dot{\boldsymbol{x}}}_{i,k}^{\mathrm{sym}} = \widehat{f}_{\mathrm{sym}}(\boldsymbol{x}_{i,k}) + \sum_{j \in \mathcal{N}_i} A_{ij} \widehat{g}_{\mathrm{sym}}(\boldsymbol{x}_{i,k}, \boldsymbol{x}_{j,k}), \tag{7}$$

where $\widehat{\dot{\boldsymbol{x}}}_{i,k}^{\mathrm{sym}} \in \mathbb{R}^d$ is the node-state derivative computed from the recovered equations. Let

$$\widehat{\dot{\boldsymbol{X}}}_k^{\mathrm{sym}} = \left[\widehat{\dot{\boldsymbol{x}}}_{1,k}^{\mathrm{sym}}, \ldots, \widehat{\dot{\boldsymbol{x}}}_{N,k}^{\mathrm{sym}}\right]^{\mathsf{T}} \in \mathbb{R}^{N \times d}.$$

We use $\widehat{\dot{\boldsymbol{X}}}_k^{\mathrm{sym}}$ as the supervision target and align the derivative $\widehat{\dot{\boldsymbol{X}}}_k$ predicted by the neural dynamics model with it through a derivative-consistency loss:

$$\mathcal{L}_{\mathrm{sym}} = \frac{1}{|\mathcal{T}|} \sum_{k \in \mathcal{T}} \mathrm{SL1}_\beta \left(\widehat{\dot{\boldsymbol{X}}}_k, \widehat{\dot{\boldsymbol{X}}}_k^{\mathrm{sym}}\right). \tag{8}$$

To avoid overly strong symbolic supervision early in feedback training, we linearly increase the symbolic feedback weight:

$$\lambda_{\mathrm{sym}}^{(e)} = \lambda_{\mathrm{sym}}^{\star} \min\left(1, \frac{e}{E_w}\right), \tag{9}$$

where e denotes the feedback-training epoch, E_w is the number of warm-up epochs and $\lambda_{\mathrm{sym}}^{\star}$ is the target symbolic feedback weight, defined as

$$\lambda_{\mathrm{sym}}^{\star} = \mathrm{clip}\left(r_{\mathrm{sym}} \frac{\mathcal{L}_{\mathrm{state}}^{(0)}}{\mathcal{L}_{\mathrm{sym}}^{(0)} + \varepsilon}, 10^{-8}, 1\right), \tag{10}$$

where $\mathcal{L}_{\mathrm{state}}^{(0)}$ and $\mathcal{L}_{\mathrm{sym}}^{(0)}$ are the state loss and derivative-consistency loss at the beginning of feedback training, respectively, $r_{\mathrm{sym}} > 0$ controls the target strength of equation derivative supervision relative to state supervision, and $\varepsilon > 0$ is a small constant for numerical stability.

The overall Stage-III objective is

$$\mathcal{L}_{\mathrm{M}_1} = \mathcal{L}_{\mathrm{state}} + \lambda_{\mathrm{roll}} \mathcal{L}_{\mathrm{roll}} + \lambda_{\mathrm{sym}}^{(e)} \mathcal{L}_{\mathrm{sym}}. \tag{11}$$

The three terms provide complementary supervision at different levels: $\mathcal{L}_{\mathrm{state}}$ promotes local state accuracy, $\mathcal{L}_{\mathrm{roll}}$ enforces temporal consistency over multi-step rollouts, and $\mathcal{L}_{\mathrm{sym}}$ promotes dynamical consistency by aligning the learned local vector field with the recovered symbolic dynamics. As analyzed in Appendix B, under the stated assumptions for explicit Euler rollouts, symbolic feedback yields a tighter worst-case rollout-error bound when the combined symbolic-model error and neural–symbolic discrepancy is smaller than the base model's vector-field error. The empirical loss $\mathcal{L}_{\mathrm{sym}}$ serves as a practical surrogate for reducing neural–symbolic mismatch on observed states, thereby supporting the proposed derivative-consistency feedback.

During training, we retain the parameters with the best validation predictive performance, yielding the final model M_1. After feedback training, we rerun the Stage-II symbolic regression procedure on M_1 to obtain the final equations used for equation-recovery evaluation.

5 Experiments

5.1 Experimental Setup

Datasets and experimental settings. We evaluate NeSyF on synthetic network dynamics and two real-world epidemic datasets. Synthetic experiments combine four network types—Barabási–Albert (BA), Watts–Strogatz (WS), Erdős–Rényi (ER), and stochastic block model (SBM) networks—with four dynamical systems: Lorenz, Rössler, Hindmarsh–Rose (HR), and Van der Pol (VdP). Real-world experiments use weekly reported influenza cases and weekly COVID-19 reports per sentinel site in Japan, with a directed weighted network constructed from inter-prefectural commuting and schooling flows. Unless otherwise specified, we set $H = 5$, $\lambda_{\mathrm{roll}} = 0.05$, $r_{\mathrm{sym}} = 0.10$, and $E_w = 150$. Full dataset construction, model architectures, and training settings are provided in Appendix D.

Baselines. For prediction, we compare with linear methods VAR (Sims, 1980) and DMD (Schmid, 2010); temporal models LSTM (Hochreiter & Schmidhuber, 1997), GRU (Cho et al., 2014), TCN (Bai et al., 2018), and Transformer (Vaswani et al., 2017); graph and spatio-temporal models GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), DCRNN (Li et al., 2018), STGCN (Yu et al., 2018), Graph WaveNet (Wu et al., 2019), MTGNN (Wu et al., 2020), and AGCRN (Bai et al., 2020); continuous-time models NeuralODE (Chen et al., 2018), NDCN (Zang & Wang, 2020), and STGODE (Fang et al., 2021); and network dynamics discovery methods LLC (Hu et al., 2025) and LaGNA (Gao et al., 2024). For equation recovery, we apply the same symbolic regression procedure to the learned dynamics of NeSyF (i.e.,M_1), LLC, and LaGNA.

Evaluation metrics. All methods are evaluated on the same test starting times $\mathcal{K}_{\mathrm{te}}$. For synthetic systems, long-horizon predictive performance is measured by the mean error over 200-step rollouts:

$$\mathrm{MAE}_{200} = \frac{1}{200|\mathcal{K}_{\mathrm{te}}|} \sum_{k \in \mathcal{K}_{\mathrm{te}}} \sum_{h=1}^{200} \frac{1}{Nd} \left\| \widehat{\boldsymbol{X}}_{k+h|k} - \boldsymbol{X}_{k+h} \right\|_1 . \tag{12}$$

Equation recovery is evaluated in terms of structural accuracy and coefficient error:

$$\begin{aligned} F_1 &= \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \\ \mathrm{MRE}_{\mathrm{coef}} &= \frac{1}{|\mathcal{S}_{\mathrm{match}}|} \sum_{q \in \mathcal{S}_{\mathrm{match}}} \frac{|\widehat{\xi}_q - \xi_q^\star|}{|\xi_q^\star| + \varepsilon}. \end{aligned} \tag{13}$$

Here, TP, FP, and FN denote correctly recovered, spurious, and missing terms, respectively, and $\mathcal{S}_{\mathrm{match}}$ is the set of matched ground-truth terms. For real-world data, where the true equations are unavailable, we report R_f^2 and R_g^2 to measure the agreement between the recovered self- and interaction-dynamics equations and the corresponding neural dynamics outputs.

5.2 Synthetic Network Experiments

State prediction. Table 1 compares the 200-step rollout performance of M_1 with representative baselines on BA and SBM networks. Three main observations follow. (1) M_1 achieves the lowest mean MAE_{200} in all 8 settings, reducing the error of LaGNA, the strongest baseline overall, by 47.1%–78.6%. (2) Structured network dynamics models outperform generic temporal, graph, and continuous-time models overall; in particular, LLC and LaGNA, which explicitly model self-dynamics and interaction dynamics, achieve substantially lower rollout errors. (3) Long-horizon rollouts reveal clear stability differences: several generic baselines accumulate substantial errors or become numerically unstable, whereas structured dynamics models remain more stable. Results on WS and ER networks and one-step prediction results on all four topologies are reported in Appendix E and show consistent trends.

Equation recovery. Table 2 compares equation recovery from the dynamics representations learned by M_1, LLC, and LaGNA using structural F_1 and coefficient error $\mathrm{MRE}_{\mathrm{coef}}$. (1) M_1 achieves the strongest and most consistent structural recovery overall, obtaining the highest F_1 in 7 of the 8 settings formed by four dynamical systems and two symbolic regressors. Its mean F_1 reaches 0.8623 with STLSQ and 0.8766 with PySR, compared with $0.1439/0.5513$ for LLC and $0.4400/0.8476$ for LaGNA, indicating that the learned dynamics of M_1 are consistently recoverable across different symbolic regressors. (2) Combining STLSQ and PySR candidates with Pareto selection further increases the mean F_1 of M_1 to 0.9174 and reduces $\mathrm{MRE}_{\mathrm{coef}}$ to 0.0930, showing that complementary symbolic search mechanisms further improve equation recovery and reduce dependence on a single regressor.

5.3 Real-World Epidemic Experiments

We first evaluate state prediction on real-world epidemic data. Since LLC and LaGNA are the only baselines that support both state prediction and explicit equation recovery, we focus on these two methods to enable a unified comparison across both tasks. Fig. 2 shows cumulative predictions under

Table 1: 200-step rollout prediction on BA and SBM networks. Values report the mean MAE_{200} over five independent runs; full mean $\pm$ sample standard deviation results are provided in Appendix E. "—" indicates numerical divergence during rollout.

	$\mathrm{MAE}_{200} \times 10^3 \downarrow$							
	BA				SBM			
Method	Lorenz	Rössler	HR	VdP	Lorenz	Rössler	HR	VdP
VAR	—	—	2665.7	131.8	—	—	1189.7	118.0
DMD	8996.5	6753.6	572.6	210.1	9175.3	5138.6	560.5	216.6
LSTM	8959.9	2961.7	688.9	372.6	9047.0	3185.4	636.7	393.5
GRU	8460.5	2888.7	687.0	339.3	9254.8	3185.6	633.2	348.2
TCN	9277.1	—	846.4	—	—	—	657.0	1753.2
Transformer	9073.1	3023.5	785.0	808.8	9202.4	3807.3	767.0	571.3
GCN	8787.4	4702.2	400.4	582.3	8911.4	2053.5	390.1	619.0
GraphSAGE	3078.0	577.6	87.3	98.5	3090.7	479.3	75.2	84.4
DCRNN	2076.3	993.6	387.3	78.6	1936.4	1139.0	288.9	63.8
STGCN	3020.0	1708.2	428.6	102.3	3038.3	1496.8	416.1	85.2
Graph WaveNet	2606.5	1278.8	400.6	226.8	2519.2	1518.5	369.3	213.2
MTGNN	3165.5	1438.2	391.0	128.2	3546.5	1205.4	478.0	118.9
AGCRN	3839.3	2605.9	1445.2	177.7	3966.0	2484.9	1049.8	152.5
NeuralODE	8262.8	2932.8	431.3	266.7	8397.8	3125.6	407.2	277.8
NDCN	4192.5	2292.3	622.4	321.2	4580.8	2311.9	530.5	321.5
STGODE	4681.8	—	2719.0	116.9	4555.5	1842.4	679.9	72.1
LLC	1833.6	139.1	10.2	19.0	1600.2	186.7	8.5	5.0
LaGNA	1095.6	51.1	10.2	1.4	1238.0	34.7	6.5	0.9
NeSyF	**580.1**	**24.6**	**3.5**	**0.3**	**605.3**	**10.7**	**2.9**	**0.3**

Table 2: Equation recovery comparison. Bold and underlined values indicate the best and second-best results, respectively, within each symbolic regressor. MRE denotes $\mathrm{MRE}_{\mathrm{coef}}$. The gray row shows M_1 after merging STLSQ and PySR candidates with Pareto selection.

		Lorenz		Rössler		HR		VdP		Avg.	
Regressor	Method	$F_1 \uparrow$	MRE↓	$F_1 \uparrow$	MRE↓	$F_1 \uparrow$	MRE↓	$F_1 \uparrow$	MRE↓	$F_1 \uparrow$	MRE↓
STLSQ	NeSyF	**0.9330**	**0.0235**	**0.9771**	**0.0055**	**0.7390**	0.2896	**0.8000**	0.3334	**0.8623**	0.1630
STLSQ	LLC	0.1087	2.4896	0.0920	0.2209	0.1185	1.5481	0.2566	0.1452	0.1439	1.1010
STLSQ	LaGNA	0.2895	0.3088	0.6344	0.0292	0.2435	**0.2803**	0.5924	**0.0004**	0.4400	**0.1547**
PySR	NeSyF	**0.9731**	0.0341	**0.9568**	**0.0057**	0.6189	0.6576	**0.9577**	0.0004	**0.8766**	0.1745
PySR	LLC	0.6100	0.4049	0.4889	0.1645	0.5604	0.4551	0.5462	0.1771	0.5513	0.3004
PySR	LaGNA	0.8928	**0.0029**	0.8941	0.0348	**0.7487**	**0.2683**	0.8549	**0.0002**	0.8476	**0.0766**
STLSQ&PySR	M_1	0.9764	0.0272	0.9871	0.0056	0.7485	0.3387	0.9577	0.0004	0.9174	0.0930

observed-state conditioning. On both the Japanese influenza and COVID-19 datasets, M_1 closely follows the observed evolution across periods of rapid growth, plateaus, and subsequent changes, whereas LLC and LaGNA exhibit more pronounced over- or underestimation in some prefectures. These results show that M_1 accurately captures epidemic state evolution on a real population-flow network.

We next examine the recovered equatio ns. As shown in Table 3, M_1 uses only 4 active terms on both datasets while maintaining high agreement with the learned neural dynamics. For influenza, it achieves $R_f^2 = 0.9998$ and $R_g^2 = 0.7264$, outperforming LLC and LaGNA while using fewer terms. For COVID-19, it obtains $R_f^2 = 0.9828$ and $R_g^2 = 0.9276$, comparable to LLC in equation fidelity while reducing the term count from 9 to 4 and substantially outperforming LaGNA. The recovered equations further suggest distinct effective interaction patterns: influenza involves nonlinear state-difference coupling through Δ_{ij} and $\exp(-|\Delta_{ij}|)$, whereas COVID-19 exhibits linear coupling

through $z_i + z_j$. Thus, M_1 provides compact explicit equations while retaining high fidelity to the learned dynamics.

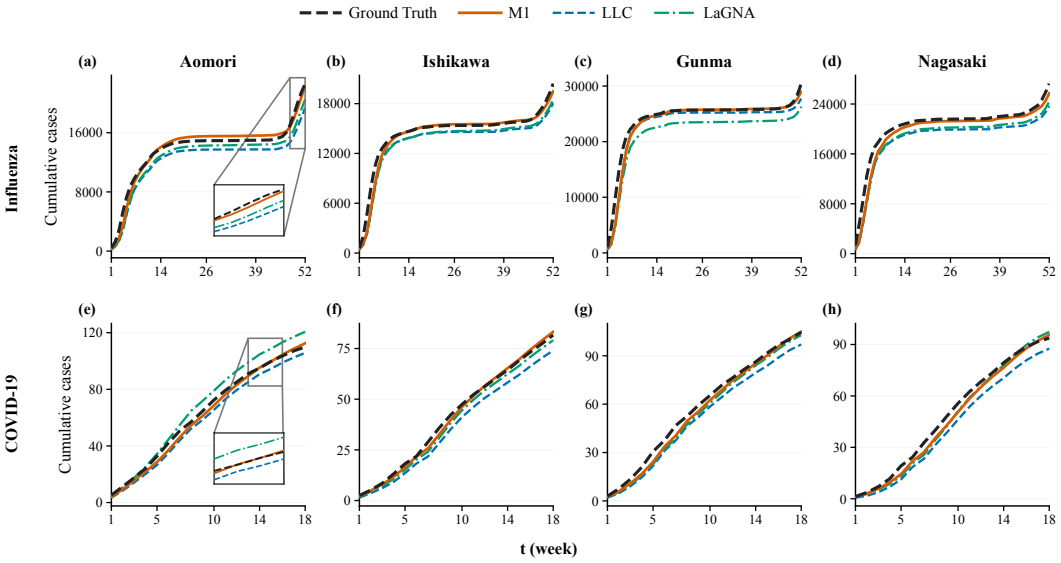


Figure 2: Real-world epidemic prediction. Influenza and COVID-19 are shown as cumulative cases and cumulative reports per sentinel site, respectively.

5.4 Ablation Study

To quantify the contribution of each component to long-horizon rollout prediction, we consider three ablations on the BA network: removing multi-step rollout supervision (w/o MS), warm-start initialization (w/o WS), or symbolic feedback (w/o SF). For w/o MS, we set $\lambda_{\mathrm{roll}} = 0$ in both Stage I and Stage III, thereby removing $\mathcal{L}_{\mathrm{roll}}$ while retaining one-step state supervision and symbolic feedback. For w/o WS, Stage III is initialized from scratch rather than from the Stage-I checkpoint M_0, while retaining the same recovered symbolic equations, training objective, and optimization budget as the full model. To isolate the effect of symbolic feedback from additional Stage-III optimization, w/o SF starts from the same M_0 checkpoint and uses the same additional optimization budget and Stage-III training protocol as M_1, with only $\mathcal{L}_{\mathrm{sym}}$ removed.

As shown in Fig. 3(a), we report the relative increase Δ in MAE_{200} of each ablation variant with respect to the full model M_1. Both w/o MS and w/o WS substantially increase long-horizon prediction error, indicating that multi-step rollout supervision and warm-start initialization both contribute to long-horizon rollout performance. Similarly, w/o SF degrades performance across all four dynamical systems, increasing MAE_{200} by approximately 70.08%, 48.53%, 117.63%, and 55.61% on Lorenz, Rössler, HR, and VdP, respectively. These results show that symbolic feedback consistently improves long-horizon prediction across the four dynamical systems, although the magnitude of the improvement varies across systems.

5.5 Hyperparameter Sensitivity

Finally, on the BA network, we examine the sensitivity of long-horizon prediction to the rollout horizon H, rollout loss weight λ_{roll}, target symbolic supervision ratio r_{sym}, and warm-up epochs E_w for symbolic feedback. As shown in Fig. 3(b)–(e), H and λ_{roll} have a stronger influence on performance, whereas r_{sym} and E_w have comparatively limited effects. For small H, MAE_{200} increases substantially on some systems, while performance becomes more stable around $H = 5$–7, with no consistent gains from further increases. Similarly, insufficient rollout-loss weight leads to larger errors on systems such as Lorenz and Rössler, whereas performance remains relatively stable over a moderate range of λ_{roll}. In contrast, varying r_{sym} causes only minor changes, and E_w has little effect over the examined range. Overall, these results indicate that a moderate rollout

Table 3: Recovered dynamical equations on two Japanese epidemic datasets. z_i denotes the normalized node state, $\Delta_{ij} = z_j - z_i$, and C_{term} counts nonzero additive terms.

	Weekly reported influenza cases in Japan			Weekly COVID-19 reports per sentinel site in Japan		
Method	Recovered equations	$C_{\text{term}} \downarrow$	$(R_f^2, R_a^2) \uparrow$	Recovered equations	$C_{\text{term}} \downarrow$	$(R_f^2, R_a^2) \uparrow$
$\mathbf{M}_1$	$f(z_i) = -0.135724 z_i,$ $g(z_i, z_j) = 0.226414 + 0.069992 \Delta_{ij}$ $- 0.231901 \exp(-\lvert\Delta_{ij}\rvert)$	**4**	$(\mathbf{0.9998}, \mathbf{0.7264})$	$f(z_i) = -0.131231 z_i,$ $g(z_i, z_j) = -0.073773 - 0.039462 z_i - 0.039462 z_j$	**4**	$(0.9828, 0.9276)$
LLC	$f(z_i) = -0.031886 - 0.063470 z_i$ $+ 0.379274 \tanh(z_i) - 0.453085 \frac{z_i}{1+\lvert z_i\rvert},$ $g(z_i, z_j) = -0.151564 - 0.075503 z_i + 0.553262 \Delta_{ij}$ $+ 0.021336 z_i^2 - 0.068908 z_j^2$ $+ 0.149839 \exp(-\lvert\Delta_{ij}\rvert)$ $- 0.117521 \frac{\Delta_{ij}}{1+\lvert z_i\rvert} - 0.705800 \frac{\Delta_{ij}}{1+\lvert z_j\rvert}$	12	$(0.9469, 0.4188)$	$f(z_i) = -0.134334 - 0.079427 z_i + 0.075854 \tanh(z_i),$ $g(z_i, z_j) = 0.047022 - 0.080327 z_j + 0.029505 z_j^2$ $+ 0.054230 \frac{\Delta_{ij}}{1+\lvert z_i\rvert} + 0.044492 \frac{\Delta_{ij}}{1+\lvert z_j\rvert}$ $+ 0.047403 \tanh(\Delta_{ij})$	9	$(\mathbf{0.9832}, \mathbf{0.9397})$
LaGNA	$f(z_i) = -0.074128 - 0.160732 z_i + 0.184149 \tanh(z_i),$ $g(z_i, z_j) = 0.076735 - 0.077046 z_i - 0.066107 z_i^2$ $- 0.023228 z_j^2 + 0.075019 z_i z_j$ $+ 0.332206 \frac{\Delta_{ij}}{1+\lvert z_i\rvert} - 1.055902 \frac{\Delta_{ij}}{1+\lvert z_j\rvert}$ $+ 0.419706 \tanh(\Delta_{ij})$	11	$(0.9794, 0.0211)$	$f(z_i) = -0.028163 z_i^2 - 0.114355 \exp(-\lvert z_i\rvert),$ $g(z_i, z_j) = 0.371748 - 0.052042 z_i + 0.059292 z_i^2$ $+ 0.059292 z_j^2 - 0.165673 z_i z_j$ $- 0.302907 \exp(-\lvert\Delta_{ij}\rvert)$ $+ 0.037658 \tanh(\Delta_{ij})$	9	$(0.7684, 0.7386)$

horizon and sufficient rollout supervision are important for stable long-horizon prediction, while the symbolic-feedback performance varies relatively little across the examined ranges of r_{sym} and E_w.

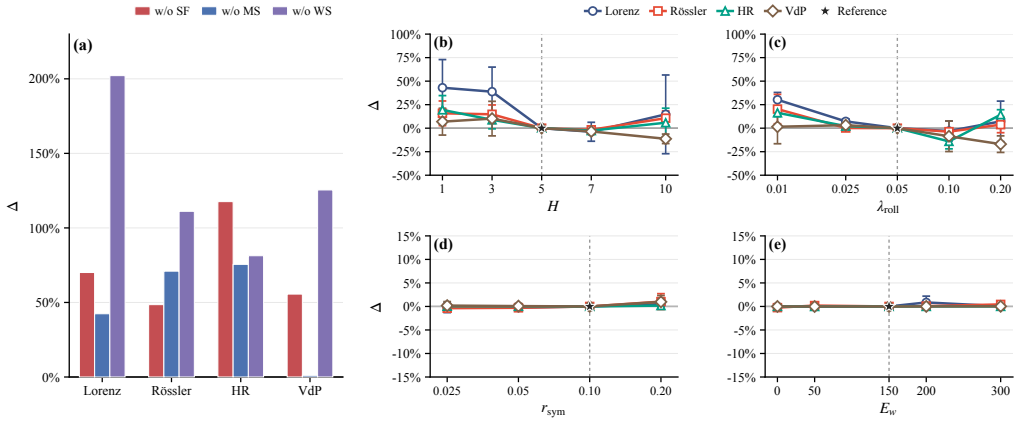


Figure 3: Ablation and hyperparameter sensitivity results on the BA network. (a) Relative increase Δ in MAE_{200} for the ablation variants with respect to the full model M_1. (b)–(e) Relative change in MAE_{200} under different H, λ_{roll}, r_{sym}, and E_w, respectively, with respect to their default settings. The x-axes of λ_{roll} and r_{sym} are shown on a logarithmic scale.

6 Conclusion and Future Work

We presented a neural-symbolic feedback framework that couples network dynamics prediction with governing equation discovery. The key idea is to move symbolic equations beyond a post-hoc readout of a trained neural model: by recovering explicit self-dynamics and interaction dynamics and feeding them back through derivative-consistency supervision, the discovered equations become structured dynamical knowledge that actively refines the predictor. Across diverse network topologies and nonlinear dynamical systems, the resulting model achieves strong long-horizon prediction while retaining dynamics that can be accurately distilled into compact explicit equations; experiments on real-world epidemic networks further demonstrate the applicability of this formulation. The accompanying error-propagation analysis relates a uniform neural–symbolic vector-field discrepancy to a worst-case Euler rollout-error bound, while the ablation results provide empirical evidence that symbolic feedback consistently improves long-horizon prediction. These findings point to a broader role for symbolic learning in scientific machine learning: explicit laws can serve not only as interpretable outputs, but also as reusable knowledge for improving data-driven models. Future work will extend this feedback paradigm to partially observed, heterogeneous, stochastic, and time-varying systems, jointly infer network topology and dynamics, and explore tighter neural-symbolic coupling in which recovered symbolic knowledge participates more continuously in model refinement.

Reproducibility Statement

We provide detailed information to facilitate reproducibility of the reported experiments. The complete neural-symbolic training procedure is described in Section 4 and summarized in Algorithm 1 in Appendix A. Appendix C details the symbolic regression candidate generation and selection procedures, including the STLSQ and PySR settings. Appendix D provides the dataset construction, network settings, ground-truth synthetic dynamics, and training hyperparameters. An anonymized implementation of the proposed method is provided in the supplementary material and will be made publicly available upon publication.

AI Use Statement

ChatGPT (OpenAI) was used to assist with translation, manuscript checking, language editing and polishing, and limited research execution, including modification and supplementation of experimental code. The core research ideas, methodology, experimental design, and code framework were developed by the authors. All AI-assisted code and text were carefully reviewed and, where applicable, tested or validated and modified by the authors. The authors take full responsibility for the accuracy, originality, and integrity of the final manuscript.

References

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17804–17815, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/ce1aad92b939420fc17005e5461e6f48-Abstract.html.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. doi: 10.48550/arXiv.1803.01271. URL https://arxiv.org/abs/1803.01271.

Baruch Barzel and Albert-László Barabási. Universality in network dynamics. *Nature Physics*, 9(10):673–681, 2013. doi: 10.1038/nphys2741.

Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and Dong-Uk Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4–5):175–308, 2006. doi: 10.1016/j.physrep.2005.10.009.

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 of the United States of America*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pp. 6571–6583, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023. doi: 10.48550/arXiv.2305.01582. URL https://arxiv.org/abs/2305.01582.

Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter W. Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17429–17442, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c9f2f917078bd2db12f23c3b413d9cba-Abstract.html.

Jiaxu Cui, Qipeng Wang, Bingyi Sun, Jiming Liu, and Bo Yang. Learning continuous network emerging dynamics from scarce observations via data-adaptive stochastic processes. *Science China Information Sciences*, 67(12):222206, 2024. doi: 10.1007/s11432-023-4216-y.

Stéphane d'Ascoli, Sören Becker, Philippe Schwaller, Alexander Mathis, and Niki Kilbertus. ODEFormer: Symbolic regression of dynamical systems with transformers. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5ed5c3c846f684a54975ad7a2525199f-Abstract-Conference.html.

Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ODE networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 364–373. ACM, 2021. doi: 10.1145/3447548.3467430.

Ting-Ting Gao and Gang Yan. Autonomous inference of complex network dynamics from incomplete and noisy data. *Nature Computational Science*, 2(3):160–168, 2022. doi: 10.1038/s43588-022-00217-0.

Ting-Ting Gao, Baruch Barzel, and Gang Yan. Learning interpretable dynamics of stochastic complex systems from experimental data. *Nature Communications*, 15:6029, 2024. doi: 10.1038/s41467-024-50378-x.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Deep generative symbolic regression. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=o7koEEMA1bR.

Jiao Hu, Jiaxu Cui, and Bo Yang. Learning interpretable network dynamics via universal neural symbolic regression. *Nature Communications*, 16:6226, 2025. doi: 10.1038/s41467-025-61575-7.

Zijie Huang, Yizhou Sun, and Wei Wang. Generalizing graph ODE for learning complex system dynamics across environments. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 798–809. ACM, 2023. doi: 10.1145/3580305.3599362.

Japan Institute for Health Security. Infectious diseases weekly report (IDWR). URL https://id-info.jihs.go.jp/en/surveillance/idwr/index.html. Historical surveillance data originally published by the National Institute of Infectious Diseases (NIID). Accessed September 6, 2026.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Ruikun Li, Huandong Wang, Jinghua Piao, Qingmin Liao, and Yong Li. Predicting long-term dynamics of complex networks via identifying skeleton in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1655–1666. ACM, 2024. doi: 10.1145/3637528.3671968.

Ruikun Li, Jingwen Cheng, Huandong Wang, Qingmin Liao, and Yong Li. Predicting the dynamics of complex system via multiscale diffusion autoencoder. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1493–1504. ACM, 2025. doi: 10.1145/3711896.3737087.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Bing Liu, Wei Luo, Gang Li, Jing Huang, and Bo Yang. Do we need an encoder-decoder to model dynamical systems on networks? In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 2178–2186. International Joint Conferences on Artificial Intelligence Organization, 2023. doi: 10.24963/ijcai.2023/242.

Ministry of Health, Labour and Welfare, Japan. Trends in reports per sentinel for influenza and COVID-19. URL https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/houkokusuunosuii_00007.html. Prefecture-level sentinel surveillance data from epidemiological week 19 of 2023 onward. Accessed September 6, 2026.

Muhammad Sahimi. Physics-informed and data-driven discovery of governing equations for complex phenomena in heterogeneous media. *Physical Review E*, 109(4):041001, 2024. doi: 10.1103/PhysRevE.109.041001.

Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. doi: 10.1017/S0022112010001217.

Christopher A. Sims. Macroeconomics and reality. *Econometrica*, 48(1):1–48, 1980. doi: 10.2307/1912017.

Statistics Bureau of Japan. 2020 population census: Tabulation on place of work or schooling for population and labour force, 2022. URL https://www.e-stat.go.jp/index.php/en/dbview?sid=0003454526. Table 5: Number of commuting employed persons and persons attending school by sex, employed persons and persons attending school, and place of working or schooling (Japan, prefectures), based on place of usual residence. Published July 22, 2022. Accessed September 6, 2026.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

Vaiva Vasiliauskaite and Nino Antulov-Fantulin. Generalization of neural network models for complex network dynamics. *Communications Physics*, 7:348, 2024. doi: 10.1038/s42005-024-01837-w.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Anima Anandkumar, Karianne Bergen, Carla P. Gomes, Shirley Ho, Pushmeet Kohli, Joan Lasenby, Jure Leskovec, Tie-Yan Liu, Arjun Manrai, Debora Marks, Bharath Ramsundar, Le Song, Jimeng Sun, Jian Tang, Petar Veličković, Max Welling, Linfeng Zhang, Connor W. Coley, Yoshua Bengio, and Marinka Zitnik. Scientific discovery in the age of artificial intelligence. *Nature*,

620(7972):47–60, 2023. doi: 10.1038/s41586-023-06221-2.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph WaveNet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 1907–1913. International Joint Conferences on Artificial Intelligence Organization, 2019. doi: 10.24963/ijcai.2019/264.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 753–763. ACM, 2020. doi: 10.1145/3394486.3403118.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3634–3640. International Joint Conferences on Artificial Intelligence Organization, 2018. doi: 10.24963/ijcai.2018/505.

Zihan Yu, Jingtao Ding, and Yong Li. Discovering network dynamics with neural symbolic regression. *Nature Computational Science*, 6(2):156–168, 2026. doi: 10.1038/s43588-025-00893-8.

Chengxi Zang and Fei Wang. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 892–902. ACM, 2020. doi: 10.1145/3394486.3403132.

A Neural-Symbolic Feedback Pipeline

The complete training procedure of the proposed neural-symbolic feedback framework is given in Algorithm 1. It consists of three stages: decoupled neural dynamics learning, symbolic equation discovery, and symbolic feedback and predictor refinement.

Algorithm 1: Neural-Symbolic Feedback Pipeline

Input: State sequence $\{\boldsymbol{X}_k\}_{k=0}^{T-1}$, adjacency matrix $\boldsymbol{A}$, sampling interval Δt

Output: Final model M_1, symbolic equations $\widehat{f}_{\mathrm{sym}}$ and $\widehat{g}_{\mathrm{sym}}$

Initialize neural parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$

`// Stage I: Decoupled Neural Dynamics Learning`

for *each Stage-I epoch* **do**

  Roll out $\widehat{\boldsymbol{X}}_{k+h|k}$ for $h = 1, \ldots, H$

  Compute $\mathcal{L}_{\mathrm{M}_0}$ using Eq. (3)

  Update $\boldsymbol{\theta}, \boldsymbol{\phi}$

$\mathrm{M}_0 \leftarrow$ best validation checkpoint

`// Stage II: Symbolic Equation Discovery`

Construct symbolic regression samples from M_0

Generate STLSQ and PySR candidates and perform Pareto selection

Obtain $\widehat{f}_{\mathrm{sym}}$ and $\widehat{g}_{\mathrm{sym}}$

`// Stage III: Symbolic Feedback and Predictor Refinement`

Initialize from M_0 and fix $\widehat{f}_{\mathrm{sym}}, \widehat{g}_{\mathrm{sym}}$

Calibrate $\lambda^{\star}_{\mathrm{sym}}$ using Eq. (10)

for $e = 1, \ldots, E_{\mathrm{M}_1}$ **do**

  Compute $\lambda^{(e)}_{\mathrm{sym}}$ using Eq. (9)

  Compute $\mathcal{L}_{\mathrm{M}_1}$ using Eq. (11)

  Update $\boldsymbol{\theta}, \boldsymbol{\phi}$

$\mathrm{M}_1 \leftarrow$ best validation checkpoint

Rerun the Stage-II symbolic regression procedure on M_1

Obtain final $\widehat{f}_{\mathrm{sym}}$ and $\widehat{g}_{\mathrm{sym}}$

B Rollout-Error Interpretation of Symbolic Feedback

Let F_0 and F_1 denote the vector fields induced by M_0 and M_1, respectively, and let $F^{\star}$ denote the underlying vector field. Let F_{sym} denote the symbolic vector field recovered from M_0 in Stage II and kept fixed during Stage-III feedback training. Over a rollout-relevant domain $\Omega \subseteq \mathbb{R}^{N \times d}$, define

$$\epsilon_0 = \sup_{\boldsymbol{X} \in \Omega} \|F_0(\boldsymbol{X}) - F^{\star}(\boldsymbol{X})\|_F, \qquad \epsilon_{\mathrm{sym}} = \sup_{\boldsymbol{X} \in \Omega} \|F_{\mathrm{sym}}(\boldsymbol{X}) - F^{\star}(\boldsymbol{X})\|_F,$$

and

$$\epsilon_{\mathrm{fb}} = \sup_{\boldsymbol{X} \in \Omega} \|F_1(\boldsymbol{X}) - F_{\mathrm{sym}}(\boldsymbol{X})\|_F.$$

We assume that these quantities are finite.

For any $\boldsymbol{X} \in \Omega$, the triangle inequality gives

$$\|F_1(\boldsymbol{X}) - F^{\star}(\boldsymbol{X})\|_F \le \|F_1(\boldsymbol{X}) - F_{\mathrm{sym}}(\boldsymbol{X})\|_F + \|F_{\mathrm{sym}}(\boldsymbol{X}) - F^{\star}(\boldsymbol{X})\|_F.$$

Taking the supremum over Ω yields

$$\sup_{\boldsymbol{X} \in \Omega} \|F_1(\boldsymbol{X}) - F^{\star}(\boldsymbol{X})\|_F \le \epsilon_{\mathrm{sym}} + \epsilon_{\mathrm{fb}}.$$

Assume that $F^{\star}$ is L-Lipschitz on Ω with respect to the Frobenius norm, i.e.,

$$\|F^{\star}(\boldsymbol{X}) - F^{\star}(\mathbf{Y})\|_F \le L\|\boldsymbol{X} - \mathbf{Y}\|_F, \qquad \forall\, \boldsymbol{X}, \mathbf{Y} \in \Omega,$$

for some $L \geq 0$. We further assume $\Delta t > 0$, that the reference rollout and the rollouts generated by both F_0 and F_1 remain in Ω, and that all rollouts start from the same initial state.

Consider the Euler-discretized reference dynamics

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \Delta t F^\star(\boldsymbol{X}_k),$$

and, for $m \in \{0, 1\}$, the corresponding model rollout

$$\widehat{\boldsymbol{X}}_{k+1}^{(m)} = \widehat{\boldsymbol{X}}_k^{(m)} + \Delta t F_m(\widehat{\boldsymbol{X}}_k^{(m)}).$$

Define

$$e_k^{(m)} = \|\widehat{\boldsymbol{X}}_k^{(m)} - \boldsymbol{X}_k\|_F.$$

For M_1, we have

$$\begin{aligned} e_{k+1}^{(1)} &\leq e_k^{(1)} + \Delta t \|F_1(\widehat{\boldsymbol{X}}_k^{(1)}) - F^\star(\boldsymbol{X}_k)\|_F \\ &\leq e_k^{(1)} + \Delta t \|F_1(\widehat{\boldsymbol{X}}_k^{(1)}) - F^\star(\widehat{\boldsymbol{X}}_k^{(1)})\|_F \\ &\quad + \Delta t \|F^\star(\widehat{\boldsymbol{X}}_k^{(1)}) - F^\star(\boldsymbol{X}_k)\|_F \\ &\leq (1 + L\Delta t) e_k^{(1)} + \Delta t(\epsilon_{\mathrm{sym}} + \epsilon_{\mathrm{fb}}). \end{aligned}$$

Since $e_0^{(1)} = 0$, iterating the recurrence for an arbitrary rollout length K gives

$$e_K^{(1)} \leq C_K(\epsilon_{\mathrm{sym}} + \epsilon_{\mathrm{fb}}),$$

where

$$C_K = \Delta t \sum_{j=0}^{K-1} (1 + L\Delta t)^j = \begin{cases} \dfrac{(1 + L\Delta t)^K - 1}{L}, & L > 0, \\ K\Delta t, & L = 0. \end{cases}$$

Applying the same argument directly to F_0 gives

$$e_K^{(0)} \leq C_K \epsilon_0.$$

Hence, a sufficient condition for M_1 to admit a tighter worst-case Euler rollout-error bound than M_0 is

$$\epsilon_{\mathrm{sym}} + \epsilon_{\mathrm{fb}} < \epsilon_0.$$

This condition highlights how symbolic accuracy and neural–symbolic consistency can jointly support a tighter worst-case rollout-error bound.

The training loss $\mathcal{L}_{\mathrm{sym}}$ in Eq. (8) measures the empirical neural–symbolic derivative discrepancy over the training states and serves as a practical surrogate for reducing this mismatch on observed states. By aligning the learned neural vector field with the recovered symbolic dynamics, it connects the feedback objective to the condition above and provides a theoretical rationale for derivative-consistency feedback under the stated assumptions.

C Symbolic Regression Candidate Generation and Selection

C.1 Symbolic Regression Candidate Generation

For multidimensional dynamics, both STLSQ and PySR are applied independently to each output component.

STLSQ. For the r-th output component, let $\boldsymbol{Z}$ denote the symbolic regression input matrix and $\boldsymbol{Y}^{(r)}$ the corresponding neural network dynamics output. STLSQ solves a ridge-regularized linear regression problem over the candidate function library $\Theta(\boldsymbol{Z})$:

$$\widehat{\boldsymbol{\xi}}^{(r)} = \arg\min_{\boldsymbol{\xi}} \left\| \boldsymbol{Y}^{(r)} - \Theta(\boldsymbol{Z})\boldsymbol{\xi} \right\|_2^2 + \alpha \|\boldsymbol{\xi}\|_2^2, \tag{14}$$

where $\boldsymbol{\xi}$ is the coefficient vector of the candidate function terms and α is the ridge regularization coefficient. STLSQ then iteratively removes terms whose absolute coefficients fall below the sparsity threshold until the support stabilizes. For the synthetic experiments, we use a polynomial function library with total degree no greater than 3. We consider $\alpha \in \{10^{-8}, 10^{-6}, 10^{-4}\}$ and $\tau \in \{10^{-6}, 3\times 10^{-6}, 10^{-5}, 3\times 10^{-5}, 10^{-4}, 3\times 10^{-4}, 10^{-3}, 3\times 10^{-3}, 10^{-2}, 3\times 10^{-2}, 10^{-1}\}$, where τ denotes the sparsity threshold.

PySR. PySR directly searches symbolic expressions over a specified operator set and produces candidates with different prediction errors and expression complexities. For the synthetic experiments, we use $\{+, -, \times\}$ as the basic binary operators. We set the number of PySR search iterations to 100, the maximum expression size to 20, and use 8 populations with 50 individuals per population. For the real-world network cases, because the unknown dynamics are not constrained by a low-order polynomial assumption, we additionally include $\exp(-|u|)$, $\tanh(u)$, and nonlinear operators such as the protected fractional form $u/(1+|v|)$ in the operator set.

C.2 Candidate Evaluation and Pareto Selection

For either the self-dynamics or interaction dynamics branch, we denote the corresponding neural network dynamics mapping by h_{nn}. Given symbolic regression samples on the validation set $\{(\boldsymbol{u}_m, \boldsymbol{y}_m)\}_{m=1}^{M}$, where $\boldsymbol{y}_m = h_{\mathrm{nn}}(\boldsymbol{u}_m)$, the validation regression error of a candidate expression $\widehat{h}$ is defined as

$$E_{\mathrm{reg}}(\widehat{h}) = \frac{1}{Md}\sum_{m=1}^{M}\left\|\widehat{h}(\boldsymbol{u}_m) - \boldsymbol{y}_m\right\|_2^2. \tag{15}$$

We denote the structural complexity of a candidate expression by $C(\widehat{h})$, defined as the total number of variables, numerical constants, and mathematical operators in the expression. For multidimensional dynamics, the validation regression error is averaged across output components, while complexity is summed over all output components. We emphasize that $C(\widehat{h})$ is used for candidate selection in Stage II, whereas C_{term} in the real-world network experiments in the main text counts only the nonzero additive terms in the self-dynamics and interaction dynamics expressions; the two quantities serve different purposes.

The candidates generated by STLSQ and PySR are first merged as

$$\mathcal{C}_h = \mathcal{C}_h^{\mathrm{STLSQ}} \cup \mathcal{C}_h^{\mathrm{PySR}}. \tag{16}$$

We then use $\big(E_{\mathrm{reg}}(\widehat{h}), C(\widehat{h})\big)$ as the minimization objectives to construct the set of Pareto-optimal candidates $\mathcal{P}_h$. If candidate $\widehat{h}_a$ is no worse than $\widehat{h}_b$ on both objectives and is strictly better on at least one, we say that $\widehat{h}_a$ dominates $\widehat{h}_b$; all candidates not dominated by any other candidate form $\mathcal{P}_h$.

Let the minimum validation regression error among the Pareto-optimal candidates be

$$E_{\mathrm{reg}}^{\min} = \min_{\widehat{h}\in\mathcal{P}_h} E_{\mathrm{reg}}(\widehat{h}). \tag{17}$$

We further select

$$\widehat{h}^{\star} = \operatorname*{arg\,min}_{\widehat{h}\in\mathcal{P}_h,\ E_{\mathrm{reg}}(\widehat{h})\le(1+\delta)E_{\mathrm{reg}}^{\min}} C(\widehat{h}), \qquad \delta = 0.05. \tag{18}$$

Thus, among candidates with validation errors close to the optimum, we select the expression with the lowest complexity. This procedure is applied separately to the self-dynamics and interaction dynamics candidates.

D Experimental Setup and Reproducibility Details

D.1 Datasets and Network Construction

Synthetic data. We use Barabási–Albert (BA), Watts–Strogatz (WS), Erdős–Rényi (ER), and stochastic block model (SBM) networks, and simulate four dynamical systems: Lorenz, Rössler,

Hindmarsh–Rose (HR), and Van der Pol (VdP). All synthetic networks contain $N = 100$ nodes. BA networks use $m = 2$; WS networks use $k = 4$ with rewiring probability $p = 0.1$; ER networks use a fixed number of edges $|\mathcal{E}| = 196$; and SBM networks contain 4 communities of size 25 with $p_{\text{in}} = 0.12$ and $p_{\text{out}} = 0.015$. The synthetic networks are undirected and unweighted. Accordingly, $\boldsymbol{A}$ is a symmetric binary adjacency matrix with $A_{ij} = 1$ if nodes i and j are connected and $A_{ij} = 0$ otherwise, and self-loops are removed. The coupling strengths are specified explicitly in the dynamical equations in Table 4.

Initial node states are sampled independently for each dynamical system. For Lorenz, all three state components are sampled from $\mathcal{U}(-10, 10)$. For Rössler, the first two components are sampled from $\mathcal{U}(-5, 5)$ and the third component from $\mathcal{U}(0, 5)$. For Hindmarsh–Rose, the three components are sampled from $\mathcal{U}(-2, 2)$, $\mathcal{U}(-10, 2)$, and $\mathcal{U}(0, 5)$, respectively. For Van der Pol, both state components are sampled from $\mathcal{U}(-2, 2)$. The ground-truth derivatives are evaluated from the dynamical equations in Table 4, and trajectories are generated using explicit Euler integration,

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \Delta t\, \dot{\boldsymbol{X}}_k, \qquad \Delta t = 0.01. \tag{19}$$

Before recording the trajectories used in the experiments, we discard an initial transient of 1000 integration steps for Lorenz, Rössler, and Van der Pol, and 5000 steps for Hindmarsh–Rose. Each recorded trajectory contains $T = 2000$ discrete network states and is split chronologically into training $[0, 1200)$, validation $[1200, 1600)$, and test $[1600, 2000)$ segments. The recorded state trajectories provide the state targets used for one-step prediction and multi-step rollout training and evaluation. The five experimental runs use random seeds 42, 43, 44, 45, and 46. For a given topology, dynamical system, and run, all compared methods use the same saved network and state trajectory.

Real-world data. We further use weekly reported influenza cases in Japan and weekly COVID-19 reports per sentinel site. The influenza data are from publicly available prefecture-level sentinel surveillance data in the Infectious Diseases Weekly Report (IDWR) (Japan Institute for Health Security), covering week 1 of 2015 through week 52 of 2019. The COVID-19 data are from publicly available prefecture-level sentinel surveillance data released by the Ministry of Health, Labour and Welfare (MHLW) (Ministry of Health, Labour and Welfare, Japan), covering week 19 of 2023 through week 14 of 2025. For both datasets, state transitions are chronologically split into 60% training, 20% validation, and 20% test sets. Node states are standardized using the mean and standard deviation computed from the training set, and we set $\Delta t = 1$ to represent one weekly interval. Missing observations are retained, with missing inputs forward-filled from previous observations and unobserved targets excluded from training and evaluation.

Both datasets use the 47 prefectures of Japan as network nodes. The inter-prefectural network is constructed from commuting and schooling origin–destination (OD) data from the 2020 Population Census released through e-Stat, Portal Site of Official Statistics of Japan (Statistics Bureau of Japan, 2022). The network is directed, with edge weights given by the inter-prefectural flow from each origin prefecture normalized by its total commuting and schooling population.

D.2 Ground-Truth Synthetic Dynamics

The ground-truth equations for the four synthetic dynamical systems are shown in Table 4.

D.3 Hyperparameter Settings

The main hyperparameter settings of our model are listed in Table 5.

D.4 Baseline Implementation and Training Settings

All prediction baselines use the same chronological data splits as the proposed method. For baselines that require a historical input sequence, the input window length is set to 12. Unless otherwise stated, trainable prediction models use a maximum training budget of 1500 epochs. All baselines are evaluated under the same experimental framework and prediction protocol. Their main model configurations and training settings are summarized in Table 6.

Table 4: Ground-truth synthetic dynamics.

System	State dimension d	Ground-truth generating equations
Lorenz	3	$\dot{x}_i = 10(y_i - x_i) + 0.1 \sum_{j=1}^{N} A_{ij} x_j,$ $\dot{y}_i = x_i(28 - z_i) - y_i,$ $\dot{z}_i = x_i y_i - \frac{8}{3} z_i.$
Rössler	3	$\dot{x}_i = -y_i - z_i + 0.1 \sum_{j=1}^{N} A_{ij}(x_j - x_i),$ $\dot{y}_i = x_i + 0.35 y_i,$ $\dot{z}_i = 0.2 + z_i(x_i - 5.7).$
Hindmarsh–Rose	3	$\dot{x}_i = y_i - x_i^3 + 3x_i^2 - z_i + 3.25 + 0.1 \sum_{j=1}^{N} A_{ij}(x_j - x_i),$ $\dot{y}_i = 1 - 5x_i^2 - y_i,$ $\dot{z}_i = 0.006\,[4(x_i + 1.6) - z_i].$
Van der Pol	2	$\dot{x}_i = y_i + 0.03 \sum_{j=1}^{N} A_{ij}(x_j - x_i),$ $\dot{y}_i = (1 - x_i^2) y_i - x_i.$

Table 5: Main training hyperparameters of the proposed model.

Setting	Value	Setting	Value
Dynamics networks	3-layer MLP	Hidden width	128
Activation	ReLU	Batch size	128
Optimizer	AdamW	Weight decay	10^{-8}
M_0 epochs	1500	M_0 initial learning rate	5×10^{-4}
M_1 epochs	1500	M_1 initial learning rate	5×10^{-5}
Rollout horizon H	5	Rollout weight λ_{roll}	0.05
Temporal decay γ	0.8	Symbolic supervision ratio r_{sym}	0.10
Feedback warm-up E_w	150		

Table 6: Main implementation and training settings of prediction baselines.

Method	Main model configuration	Optimization and training settings
VAR	Maximum lag 5; ridge regularization 10^{-6}; lag selected by BIC	Analytical fitting
DMD	SVD truncation with cumulative energy threshold 0.999	Analytical fitting
LSTM	2 recurrent layers; hidden width 128; dropout 0.1	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
GRU	2 recurrent layers; hidden width 128; dropout 0.1	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
TCN	Hidden width 128; kernel size 3; dilations $\{1, 2, 4\}$; dropout 0.1	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
Transformer	$d_{\text{model}} = 128$; 4 attention heads; 2 layers; feed-forward width 512; dropout 0.1	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
GCN	3 graph convolution layers; hidden width 128; symmetric graph normalization with self-loops	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
GraphSAGE	3 GraphSAGE layers; hidden width 128; row-normalized neighborhood aggregation	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
DCRNN	2 DCGRU layers; hidden width 64; diffusion order $K = 2$; dual random-walk diffusion	Adam, learning rate 10^{-2}, $\epsilon = 10^{-3}$; MAE loss; gradient clipping at 5
STGCN	Spatial kernel $K_s = 3$; temporal kernel $K_t = 3$; channel blocks $(d, 32, 64)$ and $(64, 32, 128)$	RMSprop, learning rate 10^{-3}; MSE loss
Graph WaveNet	4 blocks with 2 layers per block; residual/dilation channels 32; skip channels 256; end channels 512; kernel size 2; dropout 0.3	Adam, learning rate 10^{-3}, weight decay 10^{-4}; MAE loss; gradient clipping at 5
MTGNN	Graph convolution depth 2; graph top-$k = 20$; node embedding dimension 40; 3 layers; dropout 0.3	Adam, learning rate 10^{-3}, weight decay 10^{-4}; MAE loss; gradient clipping at 5
AGCRN	Recurrent hidden width 64; 2 layers; Chebyshev order $K = 2$; node embedding dimension 10	Adam, learning rate 3×10^{-3}; MAE loss
NeuralODE	Flattened network state; 3 hidden layers of width 128 with Tanh activations; RK4 integration	AdamW, learning rate 10^{-3}, weight decay 10^{-8}; Smooth L1 loss
NDCN	Hidden width 20 with normalized graph operator	Adam, learning rate 10^{-2}, weight decay 10^{-4}; MAE loss
STGODE	Physical and semantic graphs; 6 towers; 2 TCN–ODEG–TCN blocks per tower; channel widths $64/32/64$	AdamW, learning rate 2×10^{-3}; Smooth L1 loss
LLC	Neural decoupling architecture; hidden width 200	Adam, learning rate 5×10^{-4}; Smooth L1 loss; gradient clipping at 1; 1500 training epochs
LaGNA	Separate neural modules for self-dynamics, interaction dynamics, and stochastic diffusion; hidden width 100	Adam, learning rate 10^{-3}, weight decay 10^{-8}; Gaussian negative log-likelihood; OneCycle learning-rate schedule; 1500 training epochs

E Additional Experimental Results

Table 1 in the main text reports the mean 200-step rollout error MAE_{200} on BA and SBM networks. Tables 7–10 provide the corresponding full mean $\pm$ sample standard deviation results on all four network topologies over 5 independent runs. All results use the same test split and common test starting points.

Table 7: Full 200-step rollout prediction results on BA networks. Values are mean $\pm$ sample standard deviation over 5 independent runs. "—" indicates numerical divergence during rollout.

	$\mathrm{MAE}_{200} \times 10^3 \downarrow$			
Method	Lorenz	Rössler	HR	VdP
VAR	—	—	2665.7±3324.8	131.8±36.5
DMD	8996.5±412.8	6753.6±8074.4	572.6±38.3	210.1±8.4
LSTM	8959.9±212.8	2961.7±295.3	688.9±59.1	372.6±27.9
GRU	8460.5±1237.7	2888.7±325.2	687.0±80.3	339.3±25.8
TCN	9277.1±5134.3	—	846.4±218.9	—
Transformer	9073.1±140.2	3023.5±484.8	785.0±45.4	808.8±385.8
GCN	8787.4±354.4	4702.2±5662.9	400.4±39.8	582.3±55.7
GraphSAGE	3078.0±354.1	577.6±224.2	87.3±13.4	98.5±9.1
DCRNN	2076.3±217.6	993.6±509.3	387.3±63.0	78.6±6.5
STGCN	3020.0±329.0	1708.2±212.8	428.6±27.3	102.3±12.0
Graph WaveNet	2606.5±279.3	1278.8±290.6	400.6±102.8	226.8±78.8
MTGNN	3165.5±365.3	1438.2±439.7	391.0±163.1	128.2±54.9
AGCRN	3839.3±286.1	2605.9±217.2	1445.2±624.5	177.7±50.2
NeuralODE	8262.8±227.1	2932.8±342.2	431.3±29.5	266.7±18.8
NDCN	4192.5±439.7	2292.3±667.8	622.4±64.5	321.2±52.7
STGODE	4681.8±2146.9	—	2719.0±2791.3	116.9±104.6
LLC	1833.6±490.1	139.1±95.8	10.2±2.8	19.0±6.6
LaGNA	1095.6±253.4	51.1±51.4	10.2±4.2	1.4±0.6
M_1	**580.1±106.1**	**24.6±37.7**	**3.5±1.4**	**0.3±0.04**

Consistent with the BA and SBM results in the main text, M_1 achieves the lowest mean MAE_{200} across all 8 settings on WS and ER networks. Averaged over the four dynamical systems, M_1 reduces MAE_{200} relative to the best-performing baseline in each setting by approximately 64.3% on WS and 60.5% on ER. These results further indicate that the long-horizon performance gains are not limited to a particular network structure.

Let $\mathcal{K}_{\mathrm{te}}^{(1)}$ denote the set of test starting points for one-step prediction. The one-step prediction error is defined as

$$\mathrm{MAE@1} = \frac{1}{|\mathcal{K}_{\mathrm{te}}^{(1)}|} \sum_{k \in \mathcal{K}_{\mathrm{te}}^{(1)}} \frac{1}{Nd} \left\| \widehat{\boldsymbol{X}}_{k+1|k} - \boldsymbol{X}_{k+1} \right\|_1 . \tag{20}$$

Tables 11–14 report one-step prediction results on all four network topologies.

Overall, M_1 achieves the best or tied-best one-step predictive performance across different topologies and dynamical systems, showing that its long-horizon rollout advantage does not come at the cost of short-horizon state prediction accuracy. Meanwhile, methods such as LLC and LaGNA that explicitly model self-dynamics and interaction dynamics also outperform most generic temporal and graph prediction models overall. Notably, although some generic baselines achieve low one-step errors, they still exhibit substantial error accumulation during long-horizon rollout, and some methods become numerically unstable. These results show that one-step prediction accuracy alone does not fully reflect a model's ability to capture long-term dynamics.

Table 8: Full 200-step rollout prediction results on SBM networks. Values are mean ± sample standard deviation over 5 independent runs. "—" indicates numerical divergence during rollout.

	$\mathrm{MAE}_{200} \times 10^3 \downarrow$			
Method	Lorenz	Rössler	HR	VdP
VAR	—	—	1189.7±339.7	118.0±15.0
DMD	9175.3±706.5	5138.6±2378.4	560.5±49.8	216.6±18.1
LSTM	9047.0±218.2	3185.4±475.3	636.7±33.1	393.5±32.5
GRU	9254.8±339.8	3185.6±475.1	633.2±41.5	348.2±24.8
TCN	—	—	657.0±113.3	1753.2±1239.0
Transformer	9202.4±174.0	3807.3±559.0	767.0±18.5	571.3±415.9
GCN	8911.4±399.0	2053.5±425.4	390.1±50.1	619.0±39.5
GraphSAGE	3090.7±281.8	479.3±115.3	75.2±13.2	84.4±5.9
DCRNN	1936.4±94.2	1139.0±686.9	288.9±46.7	63.8±3.7
STGCN	3038.3±372.2	1496.8±269.2	416.1±41.1	85.2±13.0
Graph WaveNet	2519.2±137.1	1518.5±772.7	369.3±52.7	213.2±31.9
MTGNN	3546.5±367.6	1205.4±409.2	478.0±340.0	118.9±38.4
AGCRN	3966.0±213.1	2484.9±685.9	1049.8±582.7	152.5±12.6
NeuralODE	8397.8±230.4	3125.6±384.5	407.2±43.8	277.8±17.2
NDCN	4580.8±677.9	2311.9±493.2	530.5±97.8	321.5±52.8
STGODE	4555.5±1680.1	1842.4±1203.9	679.9±398.5	72.1±15.7
LLC	1600.2±101.8	186.7±51.5	8.5±1.7	5.0±1.9
LaGNA	1238.0±311.6	34.7±19.7	6.5±2.0	0.9±0.2
$\mathbf{M}_1$	**605.3±132.0**	**10.7±5.6**	**2.9±0.7**	**0.3±0.1**

Table 9: Full 200-step rollout prediction results on WS networks. Values are mean ± sample standard deviation over 5 independent runs. "—" indicates numerical divergence during rollout.

	$\mathrm{MAE}_{200} \times 10^3 \downarrow$			
Method	Lorenz	Rössler	HR	VdP
VAR	—	—	979.0±142.7	132.5±40.0
DMD	9439.0±441.8	4277.1±1036.7	564.6±27.9	213.4±23.8
LSTM	8981.6±169.4	3097.7±395.3	682.3±60.9	402.3±146.9
GRU	9269.5±159.6	3069.9±399.6	672.9±47.1	335.0±42.4
TCN	6987.4±32.2	—	679.2±100.2	881.0±427.2
Transformer	9139.1±65.4	3694.6±806.8	752.9±76.0	766.7±537.2
GCN	8611.8±173.5	1435.4±108.5	404.3±44.0	—
GraphSAGE	3222.3±438.4	346.2±71.5	72.4±9.6	77.1±18.6
DCRNN	1826.0±187.8	559.0±132.7	372.3±115.2	42.3±5.8
STGCN	3211.7±297.3	1426.6±121.8	433.1±21.9	97.9±23.7
Graph WaveNet	2675.8±345.8	892.5±103.7	444.5±57.9	211.9±61.3
MTGNN	3929.5±424.2	969.5±68.9	422.4±214.1	107.4±29.0
AGCRN	3975.5±269.9	2088.6±172.2	1220.1±323.4	175.3±32.8
NeuralODE	8350.4±212.7	2792.1±298.3	425.7±36.1	266.9±27.1
NDCN	4698.0±814.7	1711.9±328.7	639.6±103.1	349.0±78.1
STGODE	4318.4±2178.1	1578.4±990.7	749.2±473.6	152.9±95.6
LLC	1798.0±236.3	156.3±86.0	14.1±2.0	2.8±2.4
LaGNA	1308.0±298.9	16.3±7.1	8.7±2.2	6.7±12.8
$\mathbf{M}_1$	**686.6±184.8**	**6.2±1.2**	**3.6±0.3**	**0.3±0.1**

Table 10: Full 200-step rollout prediction results on ER networks. Values are mean ± sample standard deviation over 5 independent runs. "—" indicates numerical divergence during rollout.

	$\mathrm{MAE}_{200} \times 10^3 \downarrow$			
Method	Lorenz	Rössler	HR	VdP
VAR	—	—	1090.0±50.4	119.6±36.6
DMD	9022.7±396.8	—	591.5±40.3	210.3±9.8
LSTM	8818.6±1013.9	3086.7±502.3	646.2±47.8	362.0±16.4
GRU	8639.2±929.0	3167.6±442.6	635.4±19.4	340.4±11.3
TCN	7008.0±27.9	6310.4±5671.4	683.5±119.6	—
Transformer	9100.1±239.2	3661.6±692.1	753.6±36.8	635.4±267.6
GCN	8997.3±178.5	2448.8±662.6	385.3±48.1	626.9±19.2
GraphSAGE	3110.6±523.4	513.1±185.7	78.3±12.7	82.6±14.9
DCRNN	1967.8±153.8	1309.7±779.9	295.3±28.9	59.3±9.4
STGCN	3594.7±487.2	1694.0±403.1	427.3±33.0	91.6±12.6
Graph WaveNet	2719.5±331.4	1265.7±409.4	438.5±104.7	225.9±60.5
MTGNN	3846.7±572.5	1179.6±329.8	560.1±171.0	129.8±46.6
AGCRN	3859.2±228.7	2853.7±503.7	1338.7±443.5	144.4±16.5
NeuralODE	8493.2±142.7	3270.0±600.6	415.4±51.7	281.1±13.8
NDCN	4574.2±616.1	2039.0±701.7	490.1±160.0	307.3±78.4
STGODE	4217.6±2598.7	1747.2±1246.6	613.1±448.9	111.5±93.9
LLC	1754.0±337.8	291.6±192.6	9.2±1.4	4.3±1.0
LaGNA	1319.0±340.4	40.2±29.2	6.9±1.9	1.0±0.2
$\mathbf{M}_1$	**687.1±203.3**	**10.7±6.5**	**3.4±0.8**	**0.3±0.1**

Table 11: One-step prediction results on BA networks. Values are mean ± sample standard deviation over 5 independent runs.

	$\mathrm{MAE@1} \times 10^3 \downarrow$			
Method	Lorenz	Rössler	HR	VdP
VAR	12074.6±950.5	3119.0±415.2	391.8±32.1	55.1±3.9
DMD	5005.1±87.9	2490.9±169.6	496.4±20.0	198.4±8.3
LSTM	6814.1±118.6	2691.1±220.9	631.9±61.5	331.2±17.9
GRU	6909.8±108.0	2686.7±241.9	648.5±49.4	283.3±17.9
TCN	6963.2±349.9	2715.6±176.9	519.4±28.8	351.9±200.3
Transformer	5378.8±119.5	2519.1±158.4	587.1±30.2	138.2±6.1
GCN	379.8±10.9	29.3±11.3	4.6±0.3	8.0±0.6
GraphSAGE	36.2±3.8	6.1±2.4	1.0±0.2	1.3±0.1
DCRNN	12.7±0.8	44.4±56.3	8.1±2.6	4.9±0.9
STGCN	181.4±16.3	446.1±170.3	110.2±19.8	17.5±1.4
Graph WaveNet	53.7±4.9	56.0±61.1	10.2±1.4	9.8±2.1
MTGNN	121.9±15.0	203.3±192.0	30.7±8.9	8.9±1.6
AGCRN	57.4±3.7	187.3±64.3	47.2±14.1	8.2±0.9
NeuralODE	418.5±11.5	42.0±9.4	5.4±0.4	4.8±0.3
NDCN	66.1±12.0	85.1±66.8	10.2±2.1	6.9±1.0
STGODE	493.7±546.5	725.0±1399.0	452.0±577.0	10.9±8.2
LLC	8.2±0.5	2.8±2.5	0.2±0.05	0.3±0.1
LaGNA	6.6±0.4	1.1±0.7	0.1±0.03	0.03±0.01
$\mathbf{M}_1$	**5.2±0.6**	**0.5±0.9**	**0.1±0.03**	**0.01±0.001**

Table 12: One-step prediction results on SBM networks. Values are mean ± sample standard deviation over 5 independent runs.

	MAE@1 × 10^3 ↓			
Method	Lorenz	Rössler	HR	VdP
VAR	12126.0±755.0	3813.0±731.4	371.5±20.6	51.7±6.1
DMD	4976.3±104.4	2476.0±275.8	473.4±35.4	202.4±9.5
LSTM	6875.6±57.8	2860.4±282.0	609.2±45.0	341.4±19.3
GRU	6965.2±46.6	2855.0±258.4	585.0±43.4	289.7±17.9
TCN	6914.1±467.6	2715.6±319.3	491.3±28.0	444.5±229.3
Transformer	5257.4±149.6	2635.2±340.9	525.9±30.3	136.8±12.9
GCN	391.5±8.0	28.3±7.8	4.9±0.5	8.5±0.3
GraphSAGE	35.7±2.2	4.9±0.6	0.9±0.1	1.2±0.1
DCRNN	11.4±0.4	13.1±4.6	4.3±0.7	3.5±0.4
STGCN	190.5±23.5	315.0±79.6	117.2±34.0	16.3±1.8
Graph WaveNet	52.4±8.1	28.2±10.4	10.3±1.3	7.4±1.6
MTGNN	129.2±15.4	98.3±37.2	22.4±6.1	8.4±1.4
AGCRN	60.1±6.3	158.7±45.2	46.3±11.7	7.9±0.8
NeuralODE	415.3±9.7	40.7±5.0	5.2±0.5	4.9±0.2
NDCN	69.0±12.4	43.5±12.0	9.0±2.0	6.4±1.1
STGODE	1469.0±3091.3	1261.1±1571.2	442.7±572.0	8.5±1.9
LLC	8.2±0.6	2.2±0.5	0.2±0.04	0.1±0.02
LaGNA	7.3±0.9	0.9±0.4	0.1±0.03	0.03±0.003
$\mathbf{M}_1$	**5.4±0.6**	**0.2±0.1**	**0.1±0.01**	**0.01±0.001**

Table 13: One-step prediction results on WS networks. Values are mean ± sample standard deviation over 5 independent runs.

	MAE@1 × 10^3 ↓			
Method	Lorenz	Rössler	HR	VdP
VAR	12655.4±673.0	3020.3±206.4	406.4±40.2	43.8±8.7
DMD	5073.6±51.6	2522.8±260.0	489.7±46.1	190.0±15.6
LSTM	6914.5±45.4	2920.2±265.2	600.2±35.9	319.6±32.8
GRU	7089.0±135.7	2958.3±243.4	605.3±48.2	280.4±28.9
TCN	7158.3±78.5	2727.6±259.2	512.4±44.0	261.5±21.6
Transformer	5445.9±84.6	2738.7±245.6	544.7±52.4	126.8±12.5
GCN	402.8±1.6	20.0±0.9	5.0±0.7	9.3±0.9
GraphSAGE	36.5±3.4	3.6±0.6	0.8±0.1	1.0±0.2
DCRNN	10.2±0.2	6.5±1.3	4.4±1.2	2.9±0.1
STGCN	191.9±36.5	294.9±39.7	116.4±19.7	16.9±3.1
Graph WaveNet	56.2±9.0	22.3±4.8	10.7±1.5	8.5±0.8
MTGNN	153.0±22.1	90.1±37.6	28.7±6.2	9.0±1.0
AGCRN	60.4±4.1	115.3±20.7	39.3±7.9	7.9±0.8
NeuralODE	416.9±2.6	35.1±4.5	5.6±0.8	4.7±0.4
NDCN	59.5±12.7	35.4±2.7	10.6±1.3	5.7±1.5
STGODE	674.4±1295.4	761.4±1252.6	461.7±604.6	11.0±1.8
LLC	9.7±0.7	1.3±0.2	0.2±0.04	0.1±0.04
LaGNA	7.2±0.7	0.4±0.2	0.1±0.02	0.1±0.1
$\mathbf{M}_1$	**5.5±0.3**	**0.1±0.02**	**0.1±0.005**	**0.01±0.001**

Table 14: One-step prediction results on ER networks. Values are mean ± sample standard deviation over 5 independent runs.

	MAE@1 × 10^3 ↓			
Method	Lorenz	Rössler	HR	VdP
VAR	12461.4±352.0	4373.4±664.4	383.7±27.5	46.7±8.9
DMD	5019.1±48.3	2599.6±318.9	485.6±28.7	195.1±14.7
LSTM	6881.4±73.1	2871.1±368.5	604.4±7.1	335.7±10.4
GRU	7009.0±112.9	2916.8±399.1	617.8±29.7	294.8±12.7
TCN	7188.1±80.5	2831.7±351.0	504.1±31.6	257.6±15.5
Transformer	5318.1±44.5	2702.2±309.7	556.1±38.9	136.0±13.8
GCN	394.2±5.4	34.4±11.5	4.9±0.7	8.3±0.4
GraphSAGE	36.5±4.2	5.7±1.4	0.9±0.1	1.2±0.1
DCRNN	11.3±0.4	16.0±6.1	4.1±0.5	3.4±0.4
STGCN	228.8±37.1	421.3±137.8	109.6±12.3	16.2±1.9
Graph WaveNet	54.9±4.2	31.1±8.5	10.6±2.0	7.4±2.5
MTGNN	146.3±18.6	147.1±38.4	30.8±7.9	9.2±1.1
AGCRN	59.1±1.2	189.6±57.0	40.1±5.5	7.5±0.2
NeuralODE	418.7±6.9	44.8±10.0	5.2±0.5	4.9±0.3
NDCN	63.7±7.3	51.0±9.7	10.0±2.5	6.6±2.2
STGODE	1668.8±3005.1	799.5±1525.2	453.3±593.7	8.5±2.7
LLC	8.3±0.6	3.2±1.7	0.2±0.04	0.1±0.01
LaGNA	<u>7.1±0.6</u>	<u>1.0±0.5</u>	<u>0.1±0.03</u>	<u>0.03±0.003</u>
$\mathbf{M}_1$	**5.5±0.2**	**0.2±0.05**	**0.1±0.01**	**0.01±0.001**