Neural Dynamics on Complex Networks

Chengxi Zang, Fei Wang
Department of Healthcare Policy and Research, Weill Cornell Medicine
New York City, New York, USA
chz4001@med.cornell.edu, few2001@med.cornell.edu

Abstract

Learning dynamics on complex networks governed by differential equation systems is crucial for understanding, predicting, and controlling complex systems in science and engineering. However, this task is very challenging due to the intrinsic complexities in the structures of the high dimensional systems, their elusive continuous-time nonlinear dynamics, and their structural-dynamic dependencies. To address these challenges, we propose a differential deep learning model to learn continuous-time dynamics on complex networks in a data-driven way. We model differential equation systems by graph neural networks. Instead of mapping through a discrete number of hidden layers in the forward process, we solve the initial value problem by integrating the neural differential equation systems over time numerically. In the backward process, we learn the optimal parameters by back-propagating against the forward integration. We validate our model by learning and predicting various real-world dynamics on different complex networks in both (continuous-time) network dynamics learning setting and (regularly-sampled) structured sequence learning setting, and then apply our model to graph semi-supervised classification tasks (a one-snapshot case). The promising experimental results demonstrate our model's capability of jointly capturing the structure, dynamics, and semantics of complex systems in a unified framework.

1 Introduction

Real-world complex systems, such as brain (Gerstner et al. 2014), ecological systems (Gao, Barzel, and Barabási 2016), gene regulation (Alon 2006), human health (Bashan et al. 2016), and social networks (Zang et al. 2018), etc., are usually modeled as complex networks and their evolution are governed by some underlying nonlinear dynamics (Newman, Barabasi, and Watts 2011). Revealing such complex network dynamics is crucial for understanding the complex systems in nature. Effective analytical tools developed for this goal can further help us predict and control these complex systems.

Although the theory of (nonlinear) dynamical systems has been widely studied in different fields including applied math (Strogatz 2018),statistical physics (Newman, Barabasi, and Watts 2011), engineering (Slotine, Li, and others 1991), ecology (Gao, Barzel, and Barabási 2016) and biology (Bashan et al. 2016), these developed models are typically based on a clear knowledge of the network evolution mechanism which are thus usually referred to as *mechanistic models*. Given the complexity of the real world, there is still a large number of complex networks whose underlying dynamics are unknown yet (e.g., they can be too complex to be modeled by explicit mathematical functions). At the same time, massive data are usually generated during the evolution of these networks. Therefore, modern data-driven approaches are promising and highly demanding in learning the elusive dynamics on complex networks.

The development of a successful data-driven approach for modeling the dynamics on complex networks is very challenging: the interaction structures of the nodes in the network are complex and the number of nodes is large, which is referred to as the high-dimensionality of the complex systems; the rules governing the dynamic change of the nodes' states in complex networks are nonlinear, continuous-time and elusive; the structural and dynamic dependencies within the system are difficult to model. Recently, there is an emerging trend in the data-driven discovery of ordinary differential equations (ODEs) or partial differential equations (PDEs), including sparse regression method (Kutz et al. 2017; Mangan et al. 2016; Rudy et al. 2017), residual network (Qin, Wu, and Xiu 2018), feedforward neural network (Raissi, Perdikaris, and Karniadakis 2018), etc. However, these methods can only handle very small ODE systems or PDEs which consist of only few interaction terms. Effective learning of the dynamics on large complex networks which consist of tens of thousands of interaction terms is still largely unknown.

In this paper, we propose a differential deep learning approach to learn continuous-time dynamics on complex networks. We model (high-dimensional) differential equation systems by graph neural networks to capture the instantaneous change of network dynamics. Instead of mapping through a discrete number of layers in the forward process of the conventional neural network models (LeCun, Bengio, and Hinton 2015), we integrate the dynamics on graphs modeled by a neural differential equation system over continuous time. This is like a deep neural network with an infinite number of layers (Chen et al. 2018). In a dynamical system view, the continuous depth can be interpreted as continuous physical time, and the outputs of a hidden layer at time t are instantaneous network dynamics at that moment. In the backward learning process, we back-propagate the gradients of the su-

pervised information w.r.t. the learnable parameters against the forward integration, leading to learning the differential equation system in an end-to-end manner. Besides, we further enhance our algorithm by learning the dynamics in a hidden space learned from the original space of nodes' states. We name our model Neural Dynamics on Complex Networks (**NDCN**).

We validate our approach by three general tasks: 1) (Network dynamics learning): Can we learn the continuous-time dynamics on complex networks? 2) (Structured sequence learning): Can we predict the regularly-sampled structured sequence? (Seo et al. 2018) 3) (One-snapshot learning): Can we infer the semantic labels of nodes at the terminal time moment? The experimental results show that our model can accurately predict the real-world dynamics on various complex networks, in both continuous-time setting and regularly-sampled sequence setting. Besides, our model learns the semantic labels of nodes in the setting of graph semi-supervised learning (Kipf and Welling 2017) with very competitive performance. Our framework potentially serves as a unified framework to jointly capture the structure, dynamics, and semantics of complex systems in a data-driven manner. Our codes and datasets are open-sourced (Refer to Appendix A).

2 Related work

Dynamics of complex networks. Real-world complex systems are usually modeled as complex networks and driven by nonlinear dynamics: the dynamics of brain and human microbial are examined in (Gerstner et al. 2014) and (Bashan et al. 2016) respectively; (Gao, Barzel, and Barabási 2016) investigated the resilience dynamics of complex systems. (Barzel, Liu, and Barabási 2015) gave a pipeline to construct network dynamics. To the best of our knowledge, our NDCN model is the first neural network approach which learns continuous-time dynamics on complex networks in a data-driven manner.

Data-driven discovery of dynamics. Recently, some data-driven approaches are proposed to learn ODEs or PDEs, including sparse regression (Kutz et al. 2017), residual network (Qin, Wu, and Xiu 2018), feedforward neural network (Raissi, Perdikaris, and Karniadakis 2018), coupled neural networks (Raissi 2018) and so on. (Mangan et al. 2016) tries to learn biological networks dynamics by sparse regression over a large-enough library, which is not scalable to systems with more than 10 nodes due to the factorial growth of the complexity in the number of nodes. In all, none of them can learn the dynamics on complex systems with more than hundreds of nodes and tens of thousands of interactions.

Neural ODEs. Inspired by the residual network (He et al. 2016) and ordinary differential equation (ODE) theory (Lu et al. 2017; Ruthotto and Haber 2018), seminal work neural ODE model (Chen et al. 2018) was proposed to rewrite residual networks, normalizing flows, and recurrent neural network in a dynamical system way. However, our NDCN model deals with large and complex differential equations systems. Besides, our model solves different problems, namely learning the dynamics on complex networks.

Optimal control. Relationships between back-propagation in deep learning and optimal control theory are investigated in (Han, Li, and others 2018; Benning et al. 2019). We formulate our loss function by leveraging the concept of running loss and terminal loss in optimal control. We give novel constraints in optimal control which is modeled by neural differential equations systems on graphs. Our model solves novel tasks, e.g. learning the dynamics on complex networks and refer to Sec.3.1. and inferring the terminal labels are done by our NDCN in a data-driven manner.

Graph neural networks and temporal-GNNs. Graph neural networks (GNNs) (Wu et al. 2019b), e.g., Graph convolution network (GCN) (Kipf and Welling 2017), attention-based GNN (AGNN) (Thekumparampil et al. 2018), graph attention networks (GAT) (Veličković et al. 2017), etc., achieved state-of-the-art performance on graph semi-supervised learning tasks. However, existing GNNs usually have 1 or 2 layers and can not go deep (Li, Han, and Wu 2018; Wu et al. 2019b). Our NDCN gives a dynamical system view on GNNs: the continuous depth can be interpreted as continuous physical time, and the outputs of a hidden layer at time t are instantaneous network dynamics at that moment. By capturing continuous-time network dynamics and their transient behaviors, our model gives very competitive and even better results than above GNNs.

By combining RNNs or convolution operators with GNNs, temporal-GNNs (Yu, Yin, and Zhu 2017; Kazemi et al. 2019; Narayan and Roe 2018; Seo et al. 2018) try to predict the regularly-sampled structured sequences. However, these models can not be applied to continuous-time dynamics (observed at arbitrary physical times with different time intervals). Our NDCN not only predicts the continuous-time network dynamics at an arbitrary time or semantic labels from one snapshot but also predicts the structured sequences very well in a more succinct way with much fewer parameters.

3 Preliminaries

3.1 Problem Definition

We first introduce a differential equation system which models the dynamics on complex networks $\frac{dX(t)}{dt} = f\Big(X(t), G, W(t), t\Big)$ where $X(t) \in \mathbb{R}^{n\times d}$ represents the state (node feature values) of a dynamic system consisting of n linked nodes at time $t \in [0, \infty)$, and each node is characterized by d dimensional features. $G = (\mathcal{V}, \mathcal{E})$ is the network structure capturing how the nodes are linked to each other. $W(t)$ are parameters which control how the system evolves over time. $X(0) = X_0$ is the initial states of this system at time $t = 0$. The function $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is a function governing the dynamics of the system. In addition, nodes can have various semantic labels $Y(X, \Theta, t) \in \{0, 1\}^{n\times k}$ at time t, and Θ represents the parameters of this classification function. The problems we are trying to solve in this paper are:

- **(Network dynamics learning) How to learn the continuous-time dynamics $\frac{dX(t)}{dt}$ on complex networks from empirical data?** Mathematically, given a graph G and the observations of the states of system $\{\hat{X(t_1)}, \hat{X(t_2)}, ..., \hat{X(t_T)} | 0 \le t_1 < ... < t_T\}$, and t_1 to t_T are arbitrary physical time stamps, can we learn

differential equation systems $\frac{dX(t)}{dt} = f(X, G, W, t)$ to generate or predict continuous-time dynamics $X(t)$ at arbitrary physical time t? The arbitrary physical time means that $\{t_1, ..., t_T\}$ are possibly irregularly sampled with different observational time intervals. When $t > t_T$, we call the task **extrapolation prediction**, while $t < t_T$ and $t \neq \{t_1, ..., t_T\}$ for **interpolation prediction**.

- **(Structured sequence learning)**. As a special case when $t_1, t_2, ..., t_T$ are sampled regularly with equal time intervals, the above problem degenerates to a structured sequence learning task with an emphasis on sequential order instead of arbitrary physical time. The goal is to extrapolate next m steps' $X[t_T + 1], ..., X[t_T + m]$.
- **(One-snapshot learning) How to learn the semantic labels of $Y(X(t_T))$ at the moment $t = t_T$ for each node?** As a special case of above problem with an emphasis on a specific moment and without loss of generality, we focus on the moment at the terminal time t_T. The function Y can be a mapping from the nodes' states (e.g. humidity) to their labels (e.g. taking umbrella or not).

3.2 Network Dynamics

Let $\overrightarrow{x_i(t)} \in \mathbb{R}^{d\times 1}$ be d dimensional features of node i at time t and thus $X(t) = [\dots, \overrightarrow{x_i(t)}, \dots]^T \in \mathbb{R}^{n\times d}$. We investigate the following three real-world network dynamics and it is worth noting that the number of the interaction terms which contain $A_{i,j}$ can be very large.

- The heat diffusion dynamics $\frac{d\overrightarrow{x_i(t)}}{dt} = -k_{i,j} \sum_{j=1}^{n} A_{i,j}(\overrightarrow{x_i} - \overrightarrow{x_j})$ governed by Newton's law of cooling (Luikov 2012), which states that the rate of heat change of node i is proportional to the difference of the temperature between node i and its neighbors with heat capacity matrix A.
- The mutualistic interaction dynamics among species in ecology, governed by equation $\frac{d\overrightarrow{x_i(t)}}{dt} = b_i + \overrightarrow{x_i}(1 - \frac{\overrightarrow{x_i}}{k_i})(\frac{\overrightarrow{x_i}}{c_i} - 1) + \sum_{j=1}^{n} A_{i,j} \frac{\overrightarrow{x_i}\overrightarrow{x_j}}{d_i + e_i\overrightarrow{x_i} + h_j\overrightarrow{x_j}}$ (For brevity, the operations between vectors are element-wise). The mutualistic differential equation systems (Gao, Barzel, and Barabási 2016) capture the abundance $\vec{x_i(t)}$ of species i, consisting of incoming migration term b_i, logistic growth with population capacity k_i (Zang et al. 2018) and Allee effect (Allee et al. 1949) with cold-start threshold c_i, and mutualistic interaction term with interaction network A.
- The gene regulatory dynamics governed by Michaelis-Menten equation $\frac{d\overrightarrow{x_i(t)}}{dt} = -b_i\overrightarrow{x_i^f} + \sum_{j=1}^{n} A_{i,j} \frac{\overrightarrow{x_j^h}}{\overrightarrow{x_j^h}+1}$ where the first term models degradation when $f = 1$ or dimerization when $f = 2$, and the second term captures genetic activation tuned by the Hill coefficient h (Alon 2006; Gao, Barzel, and Barabási 2016).

Complex Networks. We consider following networks: (a) Grid network, where each node is connected with 8 neighbors ; (b) Random network, generated by Erdós and Rényi model (Erdos and Renyi 1959) ; (c) Power-law network, generated by Albert-Barabási model (Barabási and Albert 1999) ; (d) Small-world network, generated by Watts-Strogatz model (Watts and Strogatz 1998) ; and (e) Community network, generated by random partition model (Fortunato 2010) .

Visualization. To visualize dynamics on complex networks over time is not trivial. We first generate a network with n nodes by aforementioned network models. The nodes are re-ordered according to the community detection method by Newman (Newman 2010) and each node has a unique label from 1 to n. We layout these nodes on a 2-dimensional $\sqrt{n} \times \sqrt{n}$ grid and each grid point $(r, c) \in \mathbb{N}^2$ represents the i^{th} node where $i = r\sqrt{n} + c + 1$. Thus, nodes' states $X(t) \in \mathbb{R}^{n\times d}$ at time t when $d = 1$ can be visualized as a scalar field function $X : \mathbb{N}^2 \to \mathbb{R}$ over the grid. Please refer to Appendix B for the animations of these dynamics on different complex networks over time.

4 General framework

We formulate our general framework as follows:

$$\begin{aligned} \underset{W(t),\Theta(T)}{\operatorname{argmin}} \quad & \mathcal{L} = \int_0^T \mathcal{R}\Big(X(t), G, W, t\Big)\, dt + \mathcal{S}\Big(Y(X(T), \Theta)\Big) \\ \text{subject to} \quad & X_h(t) = f_e\Big(X(t)\Big) \\ & \frac{dX_h(t)}{dt} = f\Big(X_h(t), G, W, t\Big), X_h(0) \\ & X(t) = f_d\Big(X_h(t)\Big) \end{aligned} \tag{1}$$

where $\mathcal{R}(X(t), G, W, t)$ is the running loss of the dynamics on graph at time t, and $\mathcal{S}(Y(X(T), \Theta))$ is the terminal semantic loss at time T. The first constraint transforms $X(t)$ into hidden space $X_h(t)$ through encoding function f_e. The second constraint is the governing dynamics in the hidden space. The third constraint decodes the hidden signal back to the original space with decoding function f_d. The design of f_e, f, and f_d are flexible to be any neural structure (e.g. f_d is a softmax function for classification). We denote our model as *Neural Dynamics on Complex Networks* (**NDCN**).

By integrating $\frac{dX_h}{dt} = f(X_h, G, W, t)$ over time t from initial state $X_h(0)$, a.k.a. solving the initial value problem (Boyce, DiPrima, and Meade 1992) for this differential equation system, we can get the hidden continuous-time dynamics $X_h(t) = X_h(0) + \int_0^T f(X_h(\tau), G, W, \tau)\, d\tau$ and $X(t) = f_d(X_h(t))$ at arbitrary time moment $t > 0$.

We solve the initial value problem by numerical methods (e.g., 1^{st}-order Euler method, high-order method Dormand-Prince DOPRI5 (Dormand 1996), etc.). The numerical methods can approximate continuous-time dynamics $X(t)$ at arbitrary time t accurately with guaranteed error. In order to learn the learnable parameters W, we back-propagate the gradients of the loss function w.r.t the control parameters $\frac{\partial \mathcal{L}}{\partial W}$ over the numerical integration process backwards in an end-to-end manner, and solve the optimization problem by stochastic gradient descent methods (e.g., Adam (Kingma and Ba 2015)).

5 Learning continuous-time network dynamics

In this section, we investigate if our NDCN model can learn continuous-time network dynamics.

5.1 A Model Instance

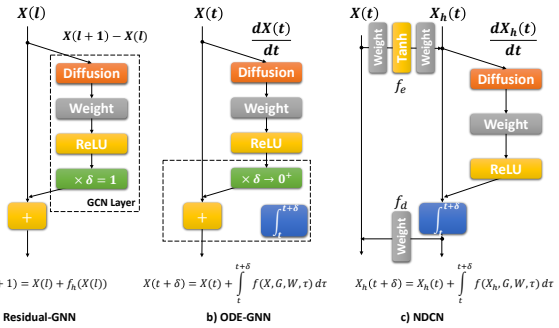

Figure 1: *Illustration of an NDCN instance.* a) Residual Graph Neural Networks, b) ODE-GNN model and c) Our Neural Dynamics on Complex Network (NDCN) model. The integer l represents the discrete l^{th} layer and the real number t represents continuous physical time.

We solve the objective function in (1) with an emphasis on running loss only. Without the loss of generality, we use ℓ_1-norm loss as the running loss $\mathcal{R}$. More concretely, we adopt two fully connected neural layers with a nonlinear hidden layer as the encoding function f_e, a graph convolution neural network (GCN) like structure (Kipf and Welling 2017) but with a different graph diffusion operator Φ to model the instantaneous network dynamics in the hidden space, and a linear decoding function f_d for regression tasks in the original signal space. Thus, our model is (see neural structure in Figure 1c) :

$$\begin{aligned} \underset{W_*,b_*}{\text{argmin}} \quad & \mathcal{L} = \int_0^T |X(t) - \hat{X(t)}|\, dt \\ \text{subject to} \quad & X_h(t) = \tanh\left(X(t)W_e + b_e\right)W_0 + b_0 \\ & \frac{dX_h(t)}{dt} = \text{ReLU}\left(\Phi X_h(t)W + b\right), X_h(0) \\ & X(t) = X_h(t)W_d + b_d \end{aligned} \tag{2}$$

where $\hat{X(t)} \in \mathbb{R}^{n\times d}$ is the supervised dynamic information available at time stamp t (in the semi-supervised case the missing information can be padded by 0). The $|\cdot|$ denotes ℓ_1-norm loss (mean element-wise absolute value difference) between $X(t)$ and $\hat{X(t)}$ at time $t \in [0, T]$. We adopt diffusion operator $\Phi = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \in \mathbb{R}^{n\times n}$ which is the normalized graph Laplacian where $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the network and $D \in \mathbb{R}^{n\times n}$ is the corresponding node degree matrix. The $W \in \mathbb{R}^{d_e\times d_e}$ and $b \in \mathbb{R}^{n\times d_e}$ are shared parameters (namely, the weights and bias of a linear connection layer) over time $t \in [0, T]$. The $W_e \in \mathbb{R}^{d\times d_e}$ and $W_0 \in \mathbb{R}^{d_e\times d_e}$ are the matrices in linear layers for encoding, while $W_d \in \mathbb{R}^{d_e\times d}$ are for decoding. The b_e, b_0, b, b_d are the biases at the corresponding layer. We lean the parameters $W_e, W_0, W, W_d, b_e, b_0, b, b_d$ from empirical data so that we can learn X in a data-driven manner.

5.2 Experiments

Baselines. To the best of our knowledge, there are no baselines for learning continuous-time dynamics on complex networks, and thus we compare the ablation models of NDCN for this task. By investigating ablation models we show that our NDCN is a minimum model for this task. We keep the loss function the same and construct the following baselines:

- The model without encoding f_e and f_d and thus no hidden space: $\frac{dX(t)}{dt} = \text{ReLU}(\Phi X(t)W + b)$, namely ODE-GNN, which learns the dynamics in the original signal space $X(t)$ as shown in Fig. 1b;
- The model without graph diffusion operator Φ: $\frac{dX_h(t)}{dt} = \text{ReLU}(X_h(t)W + b)$, i.e., an ODE Neural Network, which can be thought as a continuous-time version of forward residual neural network (See Fig. 1a and Fig. 1b for the difference between residual network and ODE network).
- The model without control parameters W: $\frac{dX_h(t)}{dt} = \text{ReLU}(\Phi X_h(t))$ which has no linear connection layer between t and $t + dt$ (where $dt \to 0$) and thus indicating a determined dynamics to spread signals.

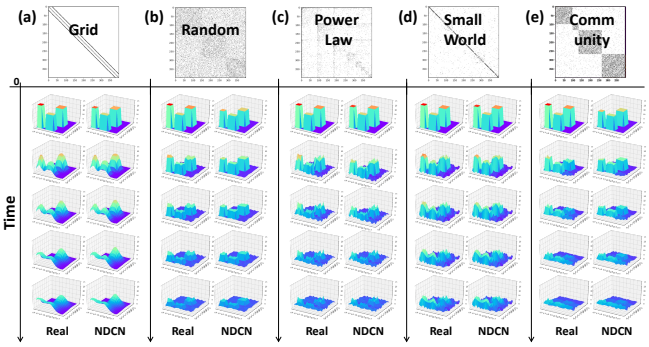

Figure 2: *Heat diffusion on different networks.* Each of the five vertical panels represents the dynamics on one network over physical time. For each network dynamics, we illustrate the sampled ground truth dynamics (left) and the dynamics generated by our NDCN (right) from top to bottom following the direction of time.

Experimental setup. We generate underlying networks with 400 nodes by network models in Sec.3.2 and the illustrations are shown in Fig. 2,10 and 11. We set the initial value $X(0)$ the same for all the experiments and thus different dynamics are only due to their different dynamic rules and underlying networks (See Appendix B). We irregularly sample 120 snapshots of the continuous-time dynamics $\{\hat{X(t_1)}, ..., \hat{X(t_{120})} | 0 \le t_1 < ... < t_{120} \le T\}$ where the time intervals between t_1, ..., t_{120} are different. We randomly choose 80 snapshots from $\hat{X(t_1)}$ to $\hat{X(t_{100})}$ for training, the left 20 snapshots from $\hat{X(t_1)}$ to $\hat{X(t_{100})}$ for testing the interpolation prediction task. We use $\hat{X(t_{101})}$ to $\hat{X(t_{120})}$ for testing the extrapolation prediction task.

We use Dormand-Prince method (Dormand 1996) to get the ground truth dynamics, and Euler method in the forward process of our NDCN (More configurations in Appendix C). We evaluate the results by the normalized ℓ_1 loss (normalized by the mean element-wise value of $\hat{X(t)}$) (See Appendix F for the ℓ_1 loss which leads to the same conclusion). Results are the mean and standard deviation of the loss over 20 inde-

pendent runs for 3 dynamic laws on 5 different networks by each method.

Results We visualize the ground-truth and learned dynamics for heat diffusion in Fig. 2. Please see the biological mutualistic dynamics and gene dynamics in Appendix Fig. 10 and 11, and please see the animations of these network dynamics in Appendix B. We find that one dynamic law may behave quite different on different networks: heat dynamics may gradually die out to be stable but follow different dynamic patterns in Fig. 2. Gene dynamics are asymptotically stable on grid in Fig. 11a but unstable on random networks in Fig. 11b or community networks in Fig. 11e. Both gene regulation dynamics in Fig. 11c and biological mutualistic dynamics in Fig. 10c show very bursty patterns on power-law networks. However, and visually speaking, our NDCN learns all these different network dynamics very well. The quantitative results of extrapolation are summarized in Table 1 (The easier task interpolation prediction shown in Table 7). We observe that our NDCN captures different dynamics on various complex networks accurately and outperforms all the continuous-time baselines by a large margin, indicating that our NDCN potentially serves as a minimum model in learning continuous-time dynamics on complex networks.

6 Learning Regularly-sampled Dynamics

What's more, our model also captures dynamics from regularly-sampled network dynamics (i.e. the structured sequence learning setting) very well.

Baselines. We compare our model with the temporal-GNN models which are usually combinations of RNN models and GNN models (Kazemi et al. 2019; Narayan and Roe 2018; Seo et al. 2018). We use GCN (Kipf and Welling 2017) as a graph structure extractor and use LSTM/GRU/RNN (Lipton, Berkowitz, and Elkan 2015) to learn the temporal relationships between ordered structured sequences. We denote each recurrent cell as LSTM/GRU/RNN and refer to Appendix E for the detailed equations. The baselines are:

- LSTM-GNN: the temporal-GNN with LSTM cell $X[t+1] = LSTM(GCN(X[t], G))$.
- GRU-GNN: the temporal-GNN with GRU cell $X[t+1] = GRU(GCN(X[t], G))$.
- RNN-GNN: the temporal-GNN with RNN cell $X[t+1] = RNN(GCN(X[t], G))$.

Experimental setup. We regularly sample 100 snapshots of the continuous-time network dynamics $\{X[\hat{t}_1], ..., X[\hat{t}_{100}] | 0 \leq t_1 < ... < t_{120} \leq T\}$ where the time intervals between t_1, ..., t_{100} are the same. We use first 80 snapshots $X[\hat{t}_1], ..., X[\hat{t}_{80}]$ for training and the left 20 snapshots $X[\hat{t}_{81}], ..., X[\hat{t}_{100}]$ for testing extrapolation prediction task. We use 5 and 10 for hidden dimension of GCN and RNN models respectively. Other settings are the same as previous continuous-time dynamics experiment.

Results We summarize the results of the extrapolation prediction of regularly-sampled dynamics in Table 2. Our NDCN predicts different dynamics on these complex networks accurately and outperforms the baselines in almost all the settings. What's more, our model capture the structure and dynamics in a much more succinct way. The learnable parameters of our NDCN , RNN-GNN, GRU-GNN, LSTM-GNN are **901**, 24530, 64770, and 84890 respectively. Our model can learn structured sequences very well with much fewer parameters than temporal-GNN models.

7 Learning semantic labels at terminal time

We investigate the third question, i.e., how to learn the semantic labels of each node at the terminal time? Various graph neural networks (GNN) (Wu et al. 2019b) achieve the state-of-the-art performance in graph semi-supervised classification task (Yang, Cohen, and Salakhutdinov 2016; Kipf and Welling 2017). Existing GNNs usually adopt 1 or 2 hidden layers and cannot go deep (Li, Han, and Wu 2018). Our framework follows the perspective of a dynamical system, and goes beyond an integer number L of hidden layers in GNNs to a real number depth t of hidden layers, implying continuous-time dynamics on the graph. Refer to Appendix G for the detailed information about the model, datasets, baselines, and their configurations.

7.1 Experiments

We validate our model in the graph semi-supervised classification setting. For the consistency of comparison with prior works, we follow the same experimental setup as (Kipf and Welling 2017; Veličković et al. 2017; Thekumparampil et al. 2018).

Results We summarize the results in Table 3. We find our NDCN outperforms many state-of-the-art GNN models.

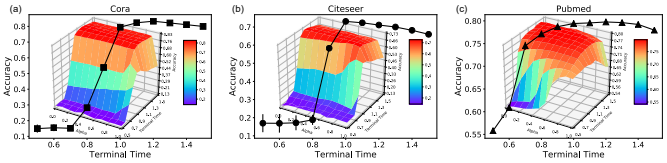

Figure 3: *Our NDCN model captures continuous-time dynamics.* Mean classification accuracy of 100 runs over terminal time when given a specific α. Insets are the accuracy over the two-dimensional space of terminal time and α

By capturing the continuous-time network dynamics to diffuse network signals, our NDCN gives better classification accuracy at terminal time $T \in \mathbb{R}^+$. Figure 3 plots the mean accuracy with error bars over terminal time T in the abovementioned α settings (we further plot the accuracy over terminal time T and α in the insets and Appendix H). We find for all the three datasets their accuracy curves follow rise and fall patterns around the best terminal time. Indeed, when the terminal time T is too small or too large, the accuracy degenerates because the features of nodes are in under-diffusion or over-diffusion states, implying the necessity in capturing continuous-time dynamics. In contrast, previous GNNs can only have an discrete number of layers which can not capture the continuous-time network dynamics accurately.

Table 1: **Continuous-time Extrapolation Prediction.** Our NDCN predicts different continuous-time network dynamics accurately. Each result is the normalized ℓ_1 error with standard deviation (in percentage $\%$) from 20 runs for 3 dynamics on 5 networks by each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	No-Encode	29.9 ± 7.3	27.8 ± 5.1	24.9 ± 5.2	24.8 ± 3.2	30.2 ± 4.4
	No-Graph	30.5 ± 1.7	5.8 ± 1.3	6.8 ± 0.5	10.7 ± 0.6	24.3 ± 3.0
	No-Control	73.4 ± 14.4	28.2 ± 4.0	25.2 ± 4.3	30.8 ± 4.7	37.1 ± 3.7
	NDCN	$\mathbf{4.1 \pm 1.2}$	$\mathbf{4.3 \pm 1.6}$	$\mathbf{4.9 \pm 0.5}$	$\mathbf{2.5 \pm 0.4}$	$\mathbf{4.8 \pm 1.0}$
Mutualistic Interaction	No-Encode	45.3 ± 3.7	9.1 ± 2.9	29.9 ± 8.8	54.5 ± 3.6	14.5 ± 5.0
	No-Graph	56.4 ± 1.1	6.7 ± 2.8	14.8 ± 6.3	54.5 ± 1.0	9.5 ± 1.5
	No-Control	140.7 ± 13.0	10.8 ± 4.3	106.2 ± 42.6	115.8 ± 12.9	16.9 ± 3.1
	NDCN	$\mathbf{26.7 \pm 4.7}$	$\mathbf{3.8 \pm 1.8}$	$\mathbf{7.4 \pm 2.6}$	$\mathbf{14.4 \pm 3.3}$	$\mathbf{3.6 \pm 1.5}$
Gene Regulation	No-Encode	31.7 ± 14.1	17.5 ± 13.0	33.7 ± 9.9	25.5 ± 7.0	26.3 ± 10.4
	No-Graph	13.3 ± 0.9	12.2 ± 0.2	43.7 ± 0.3	15.4 ± 0.3	19.6 ± 0.5
	No-Control	65.2 ± 14.2	68.2 ± 6.6	70.3 ± 7.7	58.6 ± 17.4	64.2 ± 7.0
	NDCN	$\mathbf{16.0 \pm 7.2}$	$\mathbf{1.8 \pm 0.5}$	$\mathbf{3.6 \pm 0.9}$	$\mathbf{4.3 \pm 0.9}$	$\mathbf{2.5 \pm 0.6}$

Table 2: **Regularly-sampled Extrapolation Prediction.** Our NDCN predicts different structured sequences accurately. Each result is the normalized ℓ_1 error with standard deviation (in percentage $\%$) from 20 runs for 3 dynamics on 5 networks by each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	LSTM-GNN	12.8 ± 2.1	21.6 ± 7.7	12.4 ± 5.1	11.6 ± 2.2	13.5 ± 4.2
	GRU-GNN	11.2 ± 2.2	9.1 ± 2.3	8.8 ± 1.3	9.3 ± 1.7	7.9 ± 0.8
	RNN-GNN	18.8 ± 5.9	25.0 ± 5.6	18.9 ± 6.5	21.8 ± 3.8	16.1 ± 0.0
	NDCN	$\mathbf{4.3 \pm 0.7}$	$\mathbf{4.7 \pm 1.7}$	$\mathbf{5.4 \pm 0.4}$	$\mathbf{2.7 \pm 0.4}$	$\mathbf{5.3 \pm 0.7}$
Mutualistic Interaction	LSTM-GNN	51.4 ± 3.3	24.2 ± 24.2	27.0 ± 7.1	58.2 ± 2.4	25.0 ± 22.3
	GRU-GNN	49.8 ± 4.1	$\mathbf{1.0 \pm 3.6}$	12.2 ± 0.8	51.1 ± 4.7	$\mathbf{3.7 \pm 4.0}$
	RNN-GNN	56.6 ± 0.1	8.4 ± 11.3	12.0 ± 0.4	57.4 ± 1.9	8.2 ± 6.4
	NDCN	$\mathbf{29.8 \pm 1.6}$	$\mathbf{4.7 \pm 1.1}$	$\mathbf{11.2 \pm 5.0}$	$\mathbf{15.9 \pm 2.2}$	$\mathbf{3.8 \pm 0.9}$
Gene Regulation	LSTM-GNN	27.7 ± 3.2	67.3 ± 14.2	38.8 ± 12.7	13.1 ± 2.0	53.1 ± 16.4
	GRU-GNN	24.2 ± 2.8	50.9 ± 6.4	35.1 ± 15.1	11.1 ± 1.8	46.2 ± 7.6
	RNN-GNN	28.0 ± 6.8	56.5 ± 5.7	42.0 ± 12.8	14.0 ± 5.3	46.5 ± 3.5
	NDCN	$\mathbf{18.6 \pm 9.9}$	$\mathbf{2.4 \pm 0.9}$	$\mathbf{4.1 \pm 1.4}$	$\mathbf{5.5 \pm 0.8}$	$\mathbf{2.9 \pm 0.5}$

Table 3: Test mean accuracy with standard deviation in percentage ($\%$) over 100 runs. Our NDCN model gives very competitive results compared with many GNN models.

Model	Cora	Citeseer	Pubmed
GCN	81.5	70.3	79.0
AGNN	83.1 ± 0.1	71.7 ± 0.1	$\mathbf{79.9 \pm 0.1}$
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
NDCN	$\mathbf{83.3 \pm 0.6}$	$\mathbf{73.1 \pm 0.6}$	$\mathbf{79.8 \pm 0.4}$

8 Conclusion

We propose a differential deep learning model to learn continuous-time dynamics on complex networks. We model differential equations systems by graph neural networks and integrate the neural differential equations systems over time. By capturing the continuous-time network dynamics, our NDCN gives the meanings of physical time and the continuous-time network dynamics to the depth and hidden outputs respectively, learns real-world dynamics on complex network accurately in both (irregularly-sampled) continuous-time setting and (regularly-sampled) structured sequence setting, and outperforms many GNN models in the graph semi-supervised classification task (a one-snapshot case). Codes and datasets are open-sourced at `https://github.com/calvin-zcx/NeuralDynamics`.

References

Allee, W. C.; Park, O.; Emerson, A. E.; Park, T.; Schmidt, K. P.; et al. 1949. Principles of animal ecology. Technical report, Saunders Company Philadelphia, Pennsylvania, USA.

Alon, U. 2006. *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC.

Barabási, A.-L., and Albert, R. 1999. Emergence of scaling in random networks. *science* 286(5439):509–512.

Barzel, B.; Liu, Y.-Y.; and Barabási, A.-L. 2015. Constructing minimal models for complex system dynamics. *Nature communications*.

Bashan, A.; Gibson, T. E.; Friedman, J.; Carey, V. J.; Weiss, S. T.; Hohmann, E. L.; and Liu, Y.-Y. 2016. Universality of human microbial dynamics. *Nature* 534(7606):259.

Benning, M.; Celledoni, E.; Ehrhardt, M. J.; Owren, B.; and Schönlieb, C.-B. 2019. Deep learning as optimal control problems: models and numerical methods. *arXiv preprint arXiv:1904.05657*.

Boyce, W. E.; DiPrima, R. C.; and Meade, D. B. 1992. *Elementary differential equations and boundary value problems*, volume 9. Wiley New York.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 6571–6583.

Dormand, J. R. 1996. *Numerical methods for differential equations: a computational approach*, volume 3. CRC Press.

Erdos, P., and Renyi, A. 1959. On random graphs i. *Publ. Math. Debrecen* 6:290–297.

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

Gao, J.; Barzel, B.; and Barabási, A.-L. 2016. Universal resilience patterns in complex networks. *Nature* 530(7590):307.

Gerstner, W.; Kistler, W. M.; Naud, R.; and Paninski, L. 2014. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Han, J.; Li, Q.; et al. 2018. A mean-field optimal control formulation of deep learning. *arXiv preprint arXiv:1807.01083*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2019. Relational representation learning for dynamic (knowledge) graphs: A survey. *arXiv preprint arXiv:1905.11485*.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR 2017*.

Kutz, J. N.; Rudy, S. H.; Alla, A.; and Brunton, S. L. 2017. Data-driven discovery of governing physical laws and their parametric dependencies in engineering, physics and biology. In *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 1–5. IEEE.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lipton, Z. C.; Berkowitz, J.; and Elkan, C. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Lu, Y.; Zhong, A.; Li, Q.; and Dong, B. 2017. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*.

Luikov, A. v. 2012. *Analytical heat diffusion theory*. Elsevier.

Mangan, N. M.; Brunton, S. L.; Proctor, J. L.; and Kutz, J. N. 2016. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* 2(1):52–63.

Narayan, A., and Roe, P. H. 2018. Learning graph dynamics using deep neural networks. *IFAC-PapersOnLine* 51(2):433–438.

Newman, M.; Barabasi, A.-L.; and Watts, D. J. 2011. *The structure and dynamics of networks*, volume 12. Princeton University Press.

Newman, M. 2010. *Networks: an introduction*. Oxford U. press.

Qin, T.; Wu, K.; and Xiu, D. 2018. Data driven governing equations approximation using deep neural networks. *arXiv preprint arXiv:1811.05537*.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2018. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*.

Raissi, M. 2018. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research* 19(1):932–955.

Rudy, S. H.; Brunton, S. L.; Proctor, J. L.; and Kutz, J. N. 2017. Data-driven discovery of partial differential equations. *Science Advances* 3(4):e1602614.

Ruthotto, L., and Haber, E. 2018. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*.

Seo, Y.; Defferrard, M.; Vandergheynst, P.; and Bresson, X. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, 362–373. Springer.

Slotine, J.-J. E.; Li, W.; et al. 1991. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ.

Strogatz, S. H. 2018. *Nonlinear Dynamics and Chaos with Student Solutions Manual: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press.

Thekumparampil, K. K.; Wang, C.; Oh, S.; and Li, L.-J. 2018. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Watts, D. J., and Strogatz, S. H. 1998. Collective dynamics of 'small-world'networks. *nature* 393(6684):440.

Wu, F.; Zhang, T.; Jr., A. H. S.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019a. Simplifying graph convolutional networks. *CoRR*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019b. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML 2016*, 40–48.

Yu, B.; Yin, H.; and Zhu, Z. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Zang, C.; Cui, P.; Faloutsos, C.; and Zhu, W. 2018. On power law growth of social networks. *IEEE Transactions on Knowledge and Data Engineering* 30(9):1727–1740.

A Reproducibility

To ensure the reproducibility, we open-sourced our datasets and Pytorch implementation empowered by GPU and sparse matrix at:

https://github.com/calvin-zcx/NeuralDynamics

B Animations of the real-world dynamics on different networks

Please view the animations of the three real-world dynamics on five different networks learned by different models at:

https://drive.google.com/open?id=1KBl-6Oh7BRxcQNQrPeHuKPPI6lndDa5Y

We will find our NDCN captures the real-world dynamics on different networks very accurately while the baselines can not. The detailed experimental configurations are shown as follows:

B.1 Underlying Networks

We generate various networks by as follows, and we visualize their adjacency matrix after re-ordering their nodes by the community detection method by Newman (Newman 2010).

- Grid network:

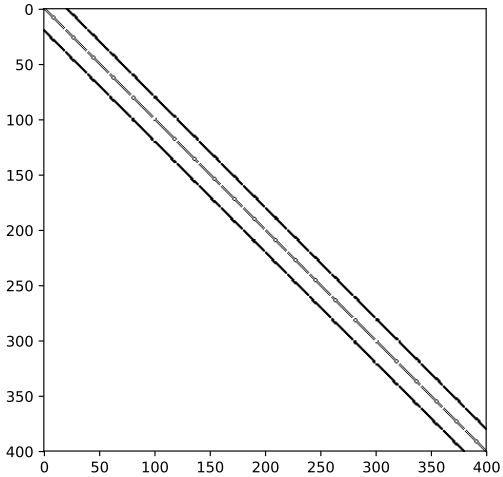

Figure 4: Adjacency matrix of grid network taking on a circulant matrix.

- Random network:

```
import networkx as nx
n = 400
G = nx.erdos_renyi_graph
(n, 0.1, seed=seed)
```

- Power-law network:

```
n = 400
G = nx.barabasi_albert_graph
(n, 5, seed=seed)
```

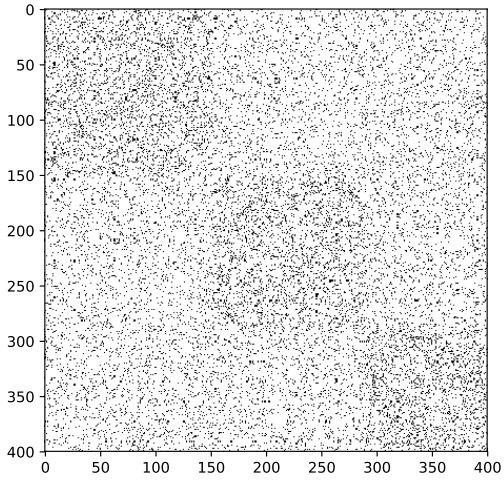

Figure 5: Adjacency matrix of random network.

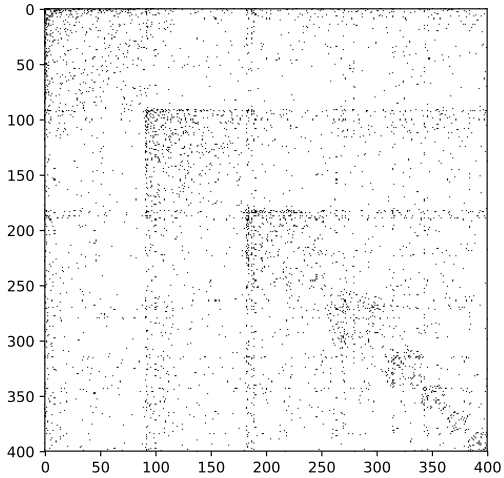

Figure 6: Adjacency matrix of power-law network.

- Small-world network:

```
n = 400
G = nx.newman_watts_strogatz_graph
(400, 5, 0.5, seed=seed)
```

- Community network:

```
n1 = int(n/3)
n2 = int(n/3)
n3 = int(n/4)
n4 = n - n1 - n2 -n3
G = nx.random_partition_graph
([n1, n2, n3, n4], .25, .01, seed=seed)
```

B.2 Initial Values of Network Dynamics

We set the initial value $X(0)$ the same for all the experimental settings and thus different dynamics are only due to their

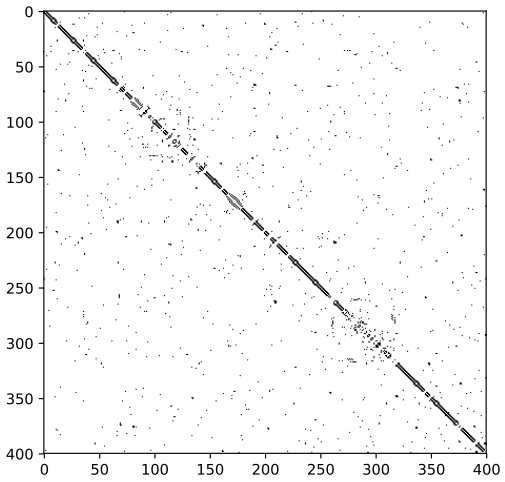

Figure 7: Adjacency matrix of small-world network.

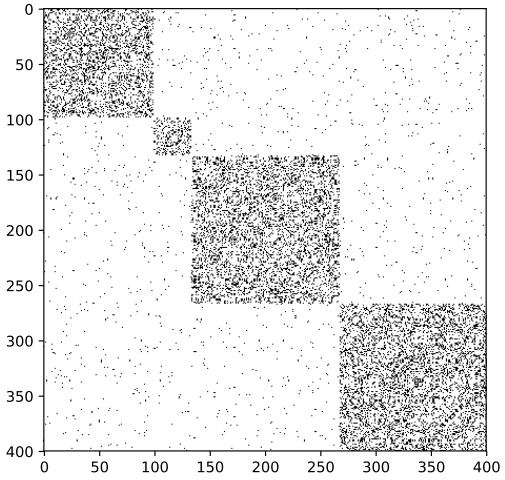

Figure 8: Adjacency matrix of community network.

different dynamic rules and underlying networks modelled by $\dot{X} = f(X, G, W, t)$ as shown in Fig. 2,**??** and **??**. Please see above animations to check out different network dynamics.

```
n = 400
N = int(np.ceil(np.sqrt(n)))
x0 = torch.zeros(N, N)
x0[int(0.05*N):int(0.25*N),
    int(0.05*N):int(0.25*N)] = 25
# x0[1:5, 1:5] = 25
for N = 20 or n= 400 case
x0[int(0.45*N):int(0.75*N),
    int(0.45*N):int(0.75*N)] = 20
# x0[9:15, 9:15] = 20 for N = 20 or n= 400 case
x0[int(0.05*N):int(0.25*N),
    int(0.35*N):int(0.65*N)] = 17
# x0[1:5, 7:13] = 17 for N = 20 or n= 400 case
```

B.3 Network Dynamics

We adopt the following three real-world dynamics from different disciplines. Please see above animations to check out the visualization of different network dynamics. The differential equation systems are shown as follows:

- The heat diffusion dynamics governed by Newton's law of cooling (Luikov 2012),

$$\frac{d\overrightarrow{x_i(t)}}{dt} = -k_{i,j} \sum_{j=1}^{n} A_{i,j}(\overrightarrow{x_i} - \overrightarrow{x_j}) \tag{3}$$

  states that the rate of heat change of node i is proportional to the difference in the temperatures between i and its neighbors with heat capacity matrix A. We use $k = 1$ here.
- The mutualistic interaction dynamics among species in ecology, governed by equation

$$\frac{d\overrightarrow{x_i(t)}}{dt} = b_i + \overrightarrow{x_i}(1 - \frac{\overrightarrow{x_i}}{k_i})(\frac{\overrightarrow{x_i}}{c_i} - 1) + \sum_{j=1}^{n} A_{i,j} \frac{\overrightarrow{x_i}\overrightarrow{x_j}}{d_i + e_i\overrightarrow{x_i} + h_j\overrightarrow{x_j}}. \tag{4}$$

  The mutualistic differential equation systems (Gao, Barzel, and Barabási 2016) capture the abundance $\vec{x_i(t)}$ of species i, consisting of incoming migration term b_i, logistic growth with population capacity k_i (Zang et al. 2018) and Allee effect (Allee et al. 1949) with cold-start threshold c_i, and mutualistic interaction term with interaction network A. We use $b = 0.1$, $k = 5.0$, $c = 1.0$, $d = 5.0$, $e = 0.9$, $h = 0.1$ here.
- The gene regulatory dynamics governed by Michaelis-Menten equation

$$\frac{d\overrightarrow{x_i(t)}}{dt} = -b_i\vec{x_i}^f + \sum_{j=1}^{n} A_{i,j} \frac{\overrightarrow{x_j}^h}{\overrightarrow{x_j}^h + 1} \tag{5}$$

  where the first term models degradation when $f = 1$ or dimerization when $f = 2$, and the second term captures genetic activation tuned by the Hill coefficient h (Gao, Barzel, and Barabási 2016). We adopt $b = 1.0$, $f = 1.0$, $h = 2.0$ here.

B.4 Terminal Time:

We use $T = 5$ for mutualistic dynamics and gene regulatory dynamics over different networks, and $T = 5, 0.1, 0.75, 2, 0.2$ for heat dynamics on the grid, random graph, power-law network, small-world network, and community network respectively due to their different time scale of network dynamics. Please see above animations to check out different network dynamics.

B.5 Visualizations of network dynamics

Please see above animations to check out the visualization of different network dynamics. We generate networks by aforementioned network models with $n = 400$ nodes. The nodes are re-ordered according to community detection method by Newman (Newman 2010). We visualize their adjacency matrices in Fig. 9,10 and 11. We layout these networks in a grid and thus nodes' states $X(t)$ are visualized as functions on the grid. Specifically, the nodes are re-ordered according to community detection method by Newman (Newman 2010) and each node has a unique label from 1 to n. We layout these nodes on a 2-dimensional $\sqrt{n} \times \sqrt{n}$ grid and each grid point $(r, c) \in \mathbb{N}^2$ represents the i^{th} node where $i = r\sqrt{n} + c + 1$. Thus, nodes' states $X(t) \in \mathbb{R}^{n \times d}$ when $d = 1$ can be visualized as a scalar field function $X : \mathbb{N}^2 \to \mathbb{R}$ over the grid.

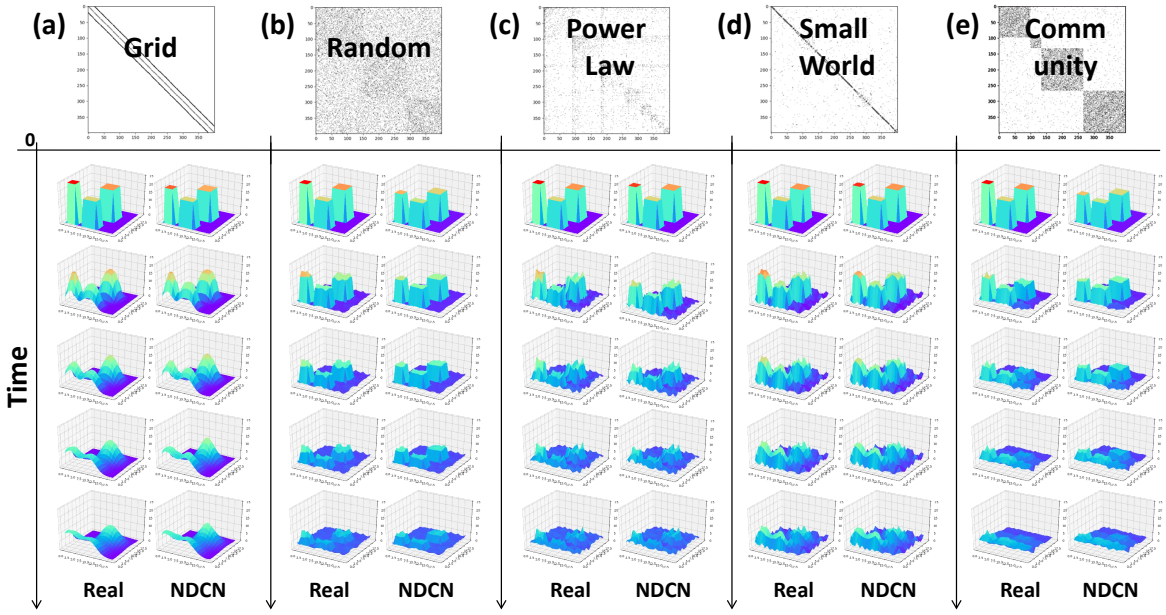

Figure 9: *Heat diffusion on different networks.* Each of the five vertical panels represents the dynamics on one network over physical time. For each network dynamics, we illustrate the sampled ground truth dynamics (left) and the dynamics generated by our NDCN (right) from top to bottom following the direction of time.

C Model configurations of Learning Network Dynamics in both continuous-time and regularly-sampled settings

We train our NDCN model by Adam (Kingma and Ba 2015). We choose 20 as the hidden dimension of $X_h \in \mathbb{R}^{n\times 20}$. We train our model for a maximum of 2000 epochs using Adam (Kingma and Ba 2015) with learning rate 0.01. We summarize our ℓ_2 regularization parameter as in Table 4 and Table 5 for Section 5 learning continuous-time network dynamics. We summarize our ℓ_2 regularization parameter as in Table 6 for Section 6 learning regularly-sampled dynamics.

D Continuous-time Interpolation Prediction

E Temporal-GNN models

We use following temporal-GNN models for structured sequence learning:

- LSTM-GNN: the temporal-GNN with LSTM cell: $X[t+1] = LSTM(GCN(X[t], G))$:

$$\begin{aligned} x_t &= ReLU(\Phi * (W_e * X[t] + b_e)) \\ i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t * c_{t-1} + i_t * g_t \\ h_t &= o_t * \tanh(c_t) \\ \hat{X[t+1]} &= W_d * h_t + b_d \end{aligned} \tag{6}$$

- GRU-GNN: the temporal-GNN with GRU cell: $X[t+1] = GRU(GCN(X[t], G))$:

$$\begin{aligned} x_t &= ReLU(\Phi * (W_e * X[t] + b_e)) \\ r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\ z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\ n_t &= \tanh(W_{in}x_t + b_{in} + r * (W_{hn}h_{t-1} + b_{hn})) \\ h_t &= (1 - z_t) * n_t + z_t * h_{t-1} \\ \hat{X[t+1]} &= W_d * h_t + b_d \end{aligned} \tag{7}$$

- RNN-GNN: the temporal-GNN with RNN cell: $X[t+1] = RNN(GCN(X[t], G))$:

$$\begin{aligned} x_t &= ReLU(\Phi * (W_e * X[t] + b_e)) \\ h_t &= \tanh(w_{ih}x_t + b_{ih} + w_{hh}h_{t-1} + b_{hh}) \\ \hat{X[t+1]} &= W_d * h_t + b_d \end{aligned} \tag{8}$$

We adopt the diffusion operator $\Phi = \tilde{D}^{-\frac{1}{2}}(\alpha I + (1-\alpha)A)\tilde{D}^{-\frac{1}{2}}$ where A is the adjacency matrix, D is the degree matrix and $\tilde{D} = \alpha I + (1-\alpha)D$ keeps Φ normalized. The differential equation system $\frac{dX}{dt} = \Phi X$ follows the dynamics of averaging the neighborhood opinion as $\frac{d\overrightarrow{x_i(t)}}{dt} = \frac{\alpha}{(1-\alpha)d_i+\alpha}\overrightarrow{x_i(t)} + \sum_j^n A_{i,j}\frac{1-\alpha}{\sqrt{(1-\alpha)d_i+\alpha}\sqrt{(1-\alpha)d_j+\alpha}}\overrightarrow{x_j(t)}$ for node i. When $\alpha = 0$, Φ averages the neighbors as normalized random walk, when $\alpha = 1$, Φ captures exponential dynamics without network effects. Here we adopt $\alpha = 0.5$, namely Φ averages both neighbors and itself as GCN in (Kipf and Welling 2017).

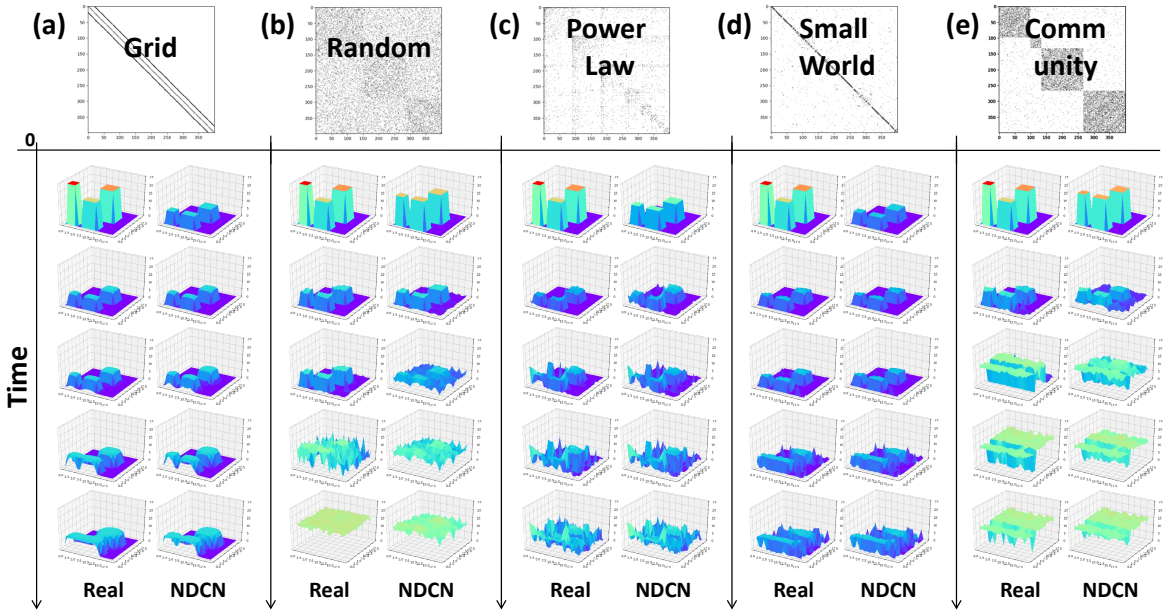

Figure 10: *Biological mutualistic interaction on different networks.*

Table 4: ℓ_2 regularization parameter configurations in continuous-time extrapolation prediction

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	No-Encode	1e-3	1e-6	1e-3	1e-3	1e-5
	No-Graph	1e-3	1e-6	1e-3	1e-3	1e-5
	No-Control	1e-3	1e-6	1e-3	1e-3	1e-5
	NDCN	1e-3	1e-6	1e-3	1e-3	1e-5
Mutualistic Interaction	No-Encode	1e-2	1e-4	1e-4	1e-4	1e-4
	No-Graph	1e-2	1e-4	1e-4	1e-4	1e-4
	No-Control	1e-2	1e-4	1e-4	1e-4	1e-4
	NDCN	1e-2	1e-4	1e-4	1e-4	1e-4
Gene Regulation	No-Embed	1e-4	1e-4	1e-4	1e-4	1e-4
	No-Graph	1e-4	1e-4	1e-4	1e-4	1e-4
	No-Control	1e-4	1e-4	1e-4	1e-4	1e-4
	NDCN	1e-4	1e-4	1e-4	1e-4	1e-4

F Results in absolute error.

We show corresponding ℓ_1 loss error in Table 8,Table 9 and Table 10 with respect to the normalized ℓ_1 loss error in Section 5 learning continuous-time network dynamics and Section 6 learning regularly-sampled dynamics. The same conclusions can be made as in Table 1,Table 7 and Table 2.

G Learning Semantic labels

We summarize datasets, baselines, and experimental setups in Section 7 learning semantic labels at the terminal time.

G.1 A Model Instance

Following the same framework as in Section 3, we propose a simple model with the terminal semantic loss $\mathcal{S}(Y(T))$ modeled by the cross-entropy loss for classification task:

$$\begin{aligned} \underset{W_e,b_e,W_d,b_d}{\operatorname{argmin}} \quad & \mathcal{L} = \int_0^T \mathcal{R}(t)\,dt - \sum_{i=1}^{n}\sum_{k=1}^{c} \hat{Y}_{i,k}(T)\log Y_{i,k}(T) \\ \text{subject to} \quad & X_h(0) = \tanh\left(X(0)W_e + b_e\right) \\ & \frac{dX_h(t)}{dt} = \mathrm{ReLU}\left(\Phi X_h(t)\right) \\ & Y(T) = \mathrm{softmax}(X_h(T)W_d + b_d) \end{aligned} \tag{9}$$

where $Y(T) \in \mathbb{R}^{n\times c}$ is the label distributions of nodes at time $T \in \mathbb{R}$ whose element $Y_{i,k}(T)$ denotes the probability of the node $i = 1,\ldots,n$ with label $k = 1,\ldots,c$ at time T. The $\hat{Y} \in \mathbb{R}^{n\times c}$ is the supervised information (again missing information can be padded by 0) observed at time $t = T$. We use differential equation system $\frac{dX(t)}{dt} = \mathrm{ReLU}(\Phi X(t))$ to spread the graph signals over continuous time $[0,T]$, i.e., $X_h(T) = X_h(0) + \int_0^T \mathrm{ReLU}\left(\Phi X_h(t)\right)\,dt$.

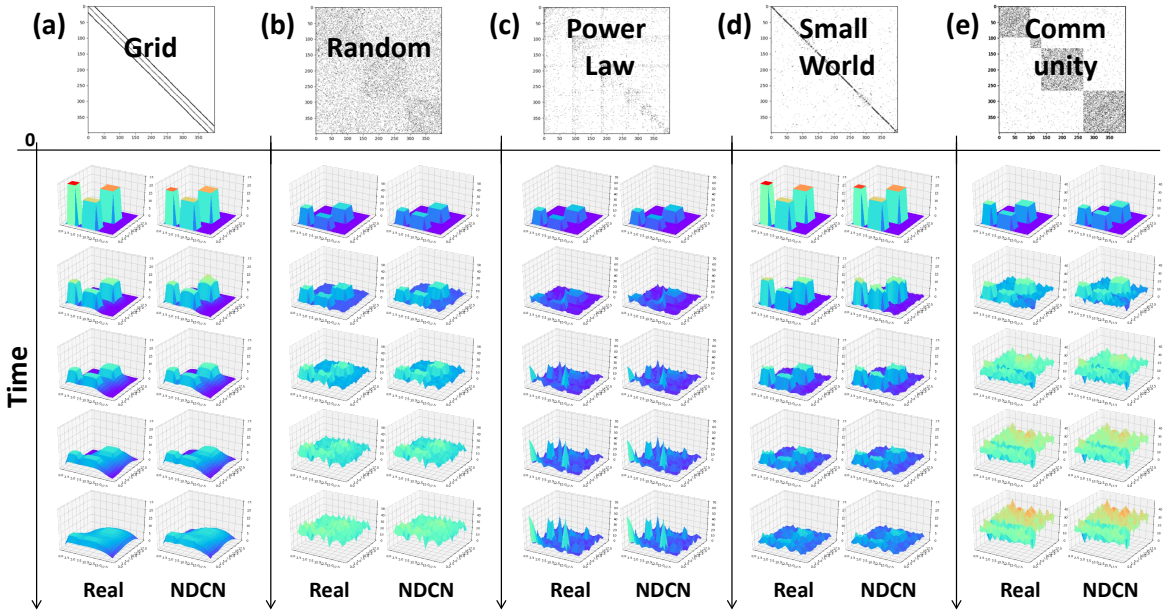

Figure 11: *Gene regulation dynamics on different networks.*

Table 5: ℓ_2 regularization parameter configurations in continuous-time interpolation prediction

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	No-Encode	1e-3	1e-6	1e-3	1e-3	1e-5
	No-Graph	1e-3	1e-6	1e-3	1e-3	1e-5
	No-Control	1e-3	1e-6	1e-3	1e-3	1e-5
	NDCN	1e-3	1e-6	1e-3	1e-3	1e-5
Mutualistic Interaction	No-Encode	1e-2	1e-4	1e-4	1e-4	1e-4
	No-Graph	1e-2	1e-4	1e-4	1e-4	1e-4
	No-Control	1e-2	1e-4	1e-4	1e-4	1e-4
	NDCN	1e-2	1e-4	1e-4	1e-4	1e-4
Gene Regulation	No-Embed	1e-4	1e-4	1e-4	1e-4	1e-4
	No-Graph	1e-4	1e-4	1e-4	1e-4	1e-4
	No-Control	1e-4	1e-4	1e-4	1e-4	1e-4
	NDCN	1e-4	1e-4	1e-4	1e-4	1e-4

We model the running loss $\int_0^T \mathcal{R}(t)\ dt$ as the ℓ_2-norm regularizer of the learnable parameters $\int_0^T \mathcal{R}(t)\ dt = \lambda(|W_e|_2^2 + |b_e|_2^2 + |W_d|_2^2 + |b_d|_2^2)$ to avoid over-fitting. We adopt the diffusion operator $\Phi = \tilde{D}^{-\frac{1}{2}}(\alpha I + (1-\alpha)A)\tilde{D}^{-\frac{1}{2}}$ where A is the adjacency matrix, D is the degree matrix and $\tilde{D} = \alpha I + (1-\alpha)D$ keeps Φ normalized. The parameter $\alpha \in [0, 1]$ tunes nodes' adherence to their previous information or their neighbors' collective opinion. We use it as a hyper-parameter here for simplicity and we can make it as a learnable parameter later.

G.2 Datasets and Baselines.

We use three standard benchmark datasets (i.e., citation network Cora, Citeseer and Pubmed), and follow the same fixed split scheme for train, validation, and test as in (Yang, Cohen, and Salakhutdinov 2016; Kipf and Welling 2017; Thekumparampil et al. 2018). We summarize the datasets in Appendix G.2 Table 11. We compare our NDCN model with graph convolution network (GCN) (Kipf and Welling 2017), attention-based graph neural network (AGNN) (Thekumparampil et al. 2018), and graph attention networks (GAT) (Veličković et al. 2017) with sophisticated attention parameters.

G.3 Experimental setup.

For the consistency of comparison with prior work, we follow the same experimental setup as (Kipf and Welling 2017; Veličković et al. 2017; Thekumparampil et al. 2018). We train our model based on the training datasets and get the accuracy of classification results from the test datasets with $1,000$ labels as summarized in Table 11. Following hyper-parameter settings apply to all the datasets. We set 16 evenly spaced time ticks in $[0, T]$ and solve the initial value problem of integrating the differential equation systems numerically by DOPRI5 (Dormand 1996). We train our model for a maximum of 100 epochs using Adam (Kingma and Ba 2015) with learning rate 0.01 and ℓ_2-norm regularization 0.024. We grid search the best terminal time $T \in [0.5, 1.5]$ and the $\alpha \in [0, 1]$. We use 256 hidden dimension. We report the mean and standard deviation of results

Table 6: ℓ_2 regularization parameter configurations in regularly-sampled extrapolation prediction

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	LSTM-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	GRU-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	RNN-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	NDCN	1e-3	1e-6	1e-3	1e-3	1e-5
Mutualistic Interaction	LSTM-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	GRU-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	RNN-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	NDCN	1e-2	1e-3	1e-4	1e-4	1e-4
Gene Regulation	LSTM-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	GRU-GNN	1e-3	1e-3	1e-3	1e-3	1e-3
	RNN-Control	1e-3	1e-3	1e-3	1e-3	1e-3
	NDCN	1e-4	1e-4	1e-4	1e-3	1e-3

Table 7: **Continuous-time Interpolation Prediction.** Our NDCN predicts different continuous-time network dynamics accurately. Each result is the normalized ℓ_1 error with standard deviation (in percentage $\%$) from 20 runs for 3 dynamics on 5 networks by each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	No-Encode	32.0 ± 12.7	26.7 ± 4.4	25.7 ± 3.8	27.9 ± 7.3	35.0 ± 6.3
	No-Graph	41.9 ± 1.8	9.4 ± 0.6	18.2 ± 1.5	25.0 ± 2.1	25.0 ± 1.4
	No-Control	56.8 ± 2.8	32.2 ± 7.0	33.5 ± 5.7	40.4 ± 3.4	39.1 ± 4.5
	NDCN	$\mathbf{3.2 \pm 0.6}$	$\mathbf{3.2 \pm 0.4}$	$\mathbf{5.6 \pm 0.6}$	$\mathbf{3.4 \pm 0.4}$	$\mathbf{4.3 \pm 0.5}$
Mutualistic Interaction	No-Encode	28.9 ± 2.0	19.9 ± 6.5	34.5 ± 13.4	27.6 ± 2.6	25.5 ± 8.7
	No-Graph	28.7 ± 4.5	7.8 ± 2.4	23.2 ± 4.2	26.9 ± 3.8	14.1 ± 2.4
	No-Control	72.2 ± 4.1	22.5 ± 10.2	63.8 ± 3.9	67.9 ± 2.9	33.9 ± 12.3
	NDCN	$\mathbf{7.6 \pm 1.1}$	$\mathbf{6.6 \pm 2.4}$	$\mathbf{6.5 \pm 1.3}$	$\mathbf{4.7 \pm 0.7}$	$\mathbf{7.9 \pm 2.9}$
Gene Regulation	No-Encode	39.2 ± 13.0	14.5 ± 12.4	33.6 ± 10.1	27.7 ± 9.4	21.2 ± 10.4
	No-Graph	25.2 ± 2.3	11.9 ± 0.2	39.4 ± 1.3	15.7 ± 0.7	18.9 ± 0.3
	No-Control	66.9 ± 8.8	31.7 ± 5.2	40.3 ± 6.6	49.0 ± 8.0	35.5 ± 5.3
	NDCN	$\mathbf{5.8 \pm 1.0}$	$\mathbf{1.5 \pm 0.6}$	$\mathbf{2.9 \pm 0.5}$	$\mathbf{4.2 \pm 0.9}$	$\mathbf{2.3 \pm 0.6}$

for 100 runs in Table 3. It's worthwhile to emphasize that in our model there is no running control parameters (i.e. linear connection layers in GNNs), no dropout (e.g., dropout rate 0.5 in GCN and 0.6 in GAT), no early stop, and no concept of layer/network depth (e.g., 2 layers in GCN and GAT).

G.4 Results

We summarize the results in Table 3. We find our NDCN outperforms many state-of-the-art GNN models. Results for the baselines are taken from (Kipf and Welling 2017; Veličković et al. 2017; Thekumparampil et al. 2018; Wu et al. 2019a). We report the mean and standard deviation of our results for 100 runs. We get our reported results in Table 3 when terminal time $T = 1.2$, $\alpha = 0$ for the Cora dataset, $T = 1.0$, $\alpha = 0.8$ for the Citeseer dataset, and $T = 1.1$, $\alpha = 0.4$ for the Pubmed dataset.

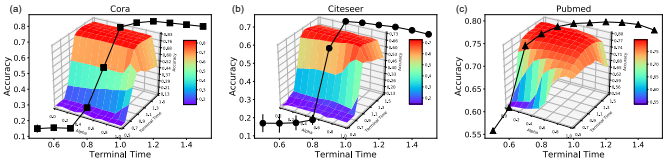

Figure 12: *Our NDCN model captures continuous-time dynamics.* Mean classification accuracy of 100 runs over terminal time when given a specific α. Insets are the accuracy over the two-dimensional space of terminal time and α

By capturing the continuous-time network dynamics to diffuse network signals, our NDCN gives better classification accuracy at terminal time $T \in \mathbb{R}^+$. Figure 3 plots the mean accuracy with error bars over terminal time T in the abovementioned α settings (we further plot the accuracy over terminal time T and α in the insets and Appendix H). We find for all the three datasets their accuracy curves follow rise and fall patterns around the best terminal time. Indeed, when the terminal time T is too small or too large, the accuracy degenerates because the features of nodes are in under-diffusion or over-diffusion states, implying the necessity in capturing continuous-time dynamics. In contrast, previous GNNs can only have an discrete number of layers which can not capture the continuous-time network dynamics accurately.

H Accuracy over terminal time and α

By capturing the continuous-time network dynamics, our NDCN gives better classification accuracy at terminal time $T \in \mathbb{R}^+$. Indeed, when the terminal time is too small or too large, the accuracy degenerates because the features of nodes are in under-diffusion or over-diffusion states. We plot the mean accuracy of 100 runs of our NDCN model over different terminal time T and α as shown in the following heatmap plots. we find for all the three datasets their accuracy curves follow rise and fall pattern around the best terminal time.

Table 8: Continuous-time Extrapolation Prediction. Our NDCN predicts different continuous-time network accurately. Each result is the ℓ_1 error with standard deviation from 20 runs for 3 dynamics on 5 networks for each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	No-Encode	1.143 ± 0.280	1.060 ± 0.195	0.950 ± 0.199	0.948 ± 0.122	1.154 ± 0.167
	No-Graph	1.166 ± 0.066	0.223 ± 0.049	0.260 ± 0.020	0.410 ± 0.023	0.926 ± 0.116
	No-Control	2.803 ± 0.549	1.076 ± 0.153	0.962 ± 0.163	1.176 ± 0.179	1.417 ± 0.140
	NDCN	$\mathbf{0.158 \pm 0.047}$	$\mathbf{0.163 \pm 0.060}$	$\mathbf{0.187 \pm 0.020}$	$\mathbf{0.097 \pm 0.016}$	$\mathbf{0.183 \pm 0.039}$
Mutualistic Interaction	No-Encode	1.755 ± 0.138	1.402 ± 0.456	2.632 ± 0.775	1.947 ± 0.106	2.007 ± 0.695
	No-Graph	2.174 ± 0.089	1.038 ± 0.434	1.301 ± 0.551	1.936 ± 0.085	1.323 ± 0.204
	No-Control	5.434 ± 0.473	1.669 ± 0.662	9.353 ± 3.751	4.111 ± 0.417	2.344 ± 0.424
	NDCN	$\mathbf{1.038 \pm 0.181}$	$\mathbf{0.584 \pm 0.277}$	$\mathbf{0.653 \pm 0.230}$	$\mathbf{0.521 \pm 0.124}$	$\mathbf{0.502 \pm 0.210}$
Gene Regulation	No-Encode	2.164 ± 0.957	6.954 ± 5.190	3.240 ± 0.954	1.445 ± 0.395	8.204 ± 3.240
	No-Graph	0.907 ± 0.058	4.872 ± 0.078	4.206 ± 0.025	0.875 ± 0.016	6.112 ± 0.143
	No-Control	4.458 ± 0.978	27.119 ± 2.608	6.768 ± 0.741	3.320 ± 0.982	20.002 ± 2.160
	NDCN	$\mathbf{1.089 \pm 0.487}$	$\mathbf{0.715 \pm 0.210}$	$\mathbf{0.342 \pm 0.088}$	$\mathbf{0.243 \pm 0.051}$	$\mathbf{0.782 \pm 0.199}$

Table 9: Continuous-time Interpolation Prediction. Our NDCN predicts different continuous-time network accurately. Each result is the ℓ_1 error with standard deviation from 20 runs for 3 dynamics on 5 networks for each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	No-Encode	1.222 ± 0.486	1.020 ± 0.168	0.982 ± 0.143	1.066 ± 0.280	1.336 ± 0.239
	No-Graph	1.600 ± 0.068	0.361 ± 0.022	0.694 ± 0.058	0.956 ± 0.079	0.954 ± 0.053
	No-Control	2.169 ± 0.108	1.230 ± 0.266	1.280 ± 0.216	1.544 ± 0.128	1.495 ± 0.171
	NDCN	$\mathbf{0.121 \pm 0.024}$	$\mathbf{0.121 \pm 0.017}$	$\mathbf{0.214 \pm 0.024}$	$\mathbf{0.129 \pm 0.017}$	$\mathbf{0.165 \pm 0.019}$
Mutualistic Interaction	No-Encode	0.620 ± 0.081	2.424 ± 0.598	1.755 ± 0.560	0.488 ± 0.077	2.777 ± 0.773
	No-Graph	0.626 ± 0.143	0.967 ± 0.269	1.180 ± 0.171	0.497 ± 0.101	1.578 ± 0.244
	No-Control	1.534 ± 0.158	2.836 ± 1.022	3.328 ± 0.314	1.212 ± 0.116	3.601 ± 0.940
	NDCN	$\mathbf{0.164 \pm 0.031}$	$\mathbf{0.843 \pm 0.267}$	$\mathbf{0.333 \pm 0.055}$	$\mathbf{0.085 \pm 0.014}$	$\mathbf{0.852 \pm 0.247}$
Gene Regulation	No-Encode	1.753 ± 0.555	4.278 ± 3.374	2.560 ± 0.765	1.180 ± 0.389	5.106 ± 2.420
	No-Graph	1.140 ± 0.101	3.768 ± 0.316	3.137 ± 0.264	0.672 ± 0.050	4.639 ± 0.399
	No-Control	3.010 ± 0.228	9.939 ± 1.185	3.139 ± 0.313	2.082 ± 0.293	8.659 ± 0.952
	NDCN	$\mathbf{0.262 \pm 0.046}$	$\mathbf{0.455 \pm 0.174}$	$\mathbf{0.222 \pm 0.034}$	$\mathbf{0.180 \pm 0.032}$	$\mathbf{0.562 \pm 0.130}$

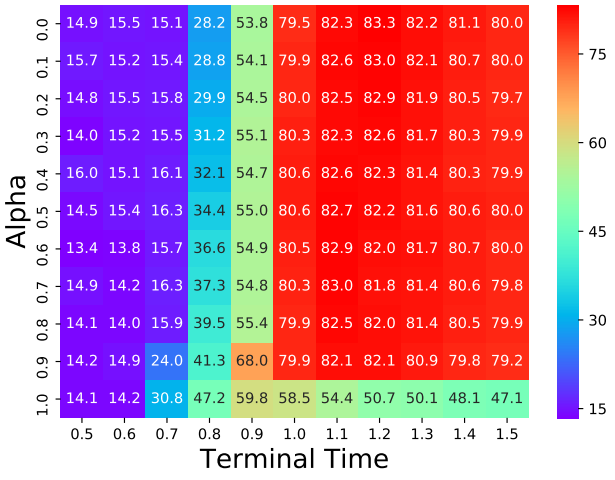

Figure 13: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Cora dataset in heatmap plot.

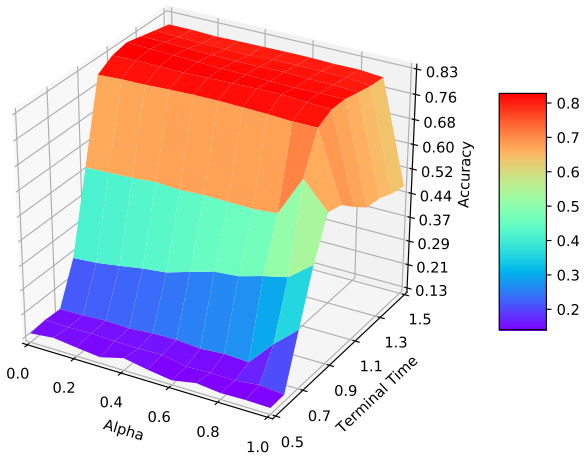

Figure 14: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Cora dataset in 3D surface plot.

Table 10: Regularly-sampled Extrapolation Prediction. Our NDCN predicts different structured sequences accurately. Each result is the ℓ_1 error with standard deviation from 20 runs for 3 dynamics on 5 networks for each method.

		Grid	Random	Power Law	Small World	Community
Heat Diffusion	LSTM-GNN	0.489 ± 0.081	0.824 ± 0.294	0.475 ± 0.196	0.442 ± 0.083	0.517 ± 0.162
	GRU-GNN	0.428 ± 0.085	0.349 ± 0.090	0.337 ± 0.049	0.357 ± 0.065	0.302 ± 0.031
	RNN-GNN	0.717 ± 0.227	0.957 ± 0.215	0.722 ± 0.247	0.833 ± 0.145	0.615 ± 0.000
	NDCN	$\mathbf{0.165 \pm 0.027}$	$\mathbf{0.180 \pm 0.063}$	$\mathbf{0.208 \pm 0.015}$	$\mathbf{0.103 \pm 0.014}$	$\mathbf{0.201 \pm 0.029}$
Mutualistic Interaction	LSTM-GNN	1.966 ± 0.126	3.749 ± 3.749	2.380 ± 0.626	2.044 ± 0.086	3.463 ± 3.095
	GRU-GNN	1.905 ± 0.157	$\mathbf{0.162 \pm 0.564}$	1.077 ± 0.071	1.792 ± 0.165	$\mathbf{0.510 \pm 0.549}$
	RNN-GNN	2.165 ± 0.004	1.303 ± 1.747	1.056 ± 0.034	2.012 ± 0.065	1.140 ± 0.887
	NDCN	$\mathbf{1.414 \pm 0.060}$	$\mathbf{0.734 \pm 0.168}$	$\mathbf{0.990 \pm 0.442}$	$\mathbf{0.557 \pm 0.078}$	$\mathbf{0.528 \pm 0.122}$
Gene Regulation	LSTM-GNN	1.883 ± 0.218	26.750 ± 5.634	3.733 ± 1.220	0.743 ± 0.112	16.534 ± 5.094
	GRU-GNN	1.641 ± 0.191	20.240 ± 2.549	3.381 ± 1.455	0.626 ± 0.099	14.4 ± 2.358
	RNN-GNN	1.906 ± 0.464	22.46 ± 2.276	4.036 ± 1.229	0.795 ± 0.300	14.496 ± 1.077
	NDCN	$\mathbf{1.267 \pm 0.672}$	$\mathbf{0.946 \pm 0.357}$	$\mathbf{0.397 \pm 0.133}$	$\mathbf{0.312 \pm 0.043}$	$\mathbf{0.901 \pm 0.160}$

Table 11: Statistics for three real-world citation network datasets. N, E, D, C represent number of nodes, edges, features, classes respectively.

Dataset	N	E	D	C	Train/Valid/Test
Cora	2,708	5,429	1,433	7	140/500/1,000
Citeseer	3,327	4,732	3,703	6	120/500/1,000
Pubmed	19,717	44,338	500	3	60/500/1,000

Table 12: Test mean accuracy with standard deviation in percentage (%) over 100 runs. Our NDCN model gives very competitive results compared with many GNN models.

Model	Cora	Citeseer	Pubmed
GCN	81.5	70.3	79.0
AGNN	83.1 ± 0.1	71.7 ± 0.1	$\mathbf{79.9 \pm 0.1}$
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
NDCN	$\mathbf{83.3 \pm 0.6}$	$\mathbf{73.1 \pm 0.6}$	$\mathbf{79.8 \pm 0.4}$

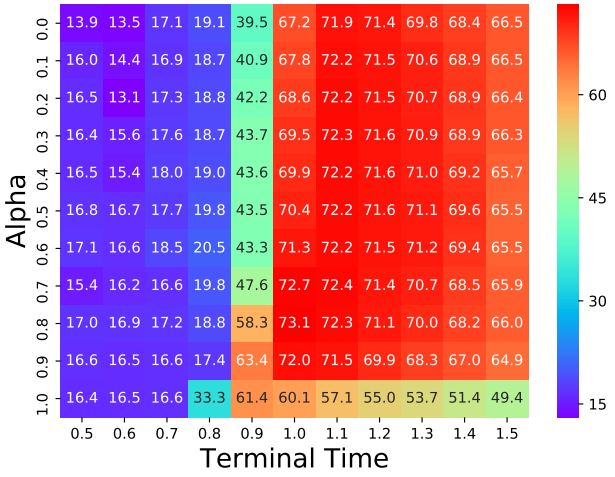

Figure 15: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Citeseer dataset.

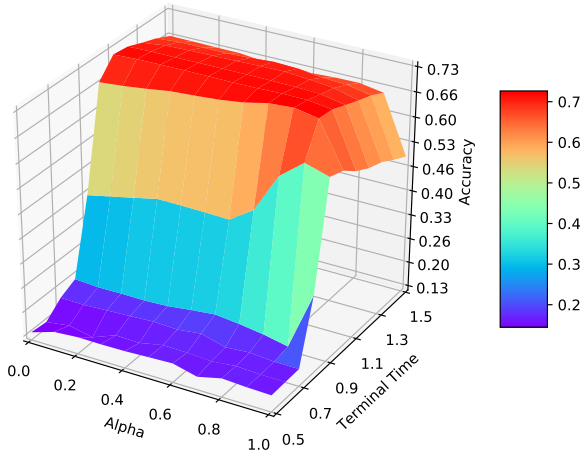

Figure 16: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Citeseer dataset in 3D surface plot.

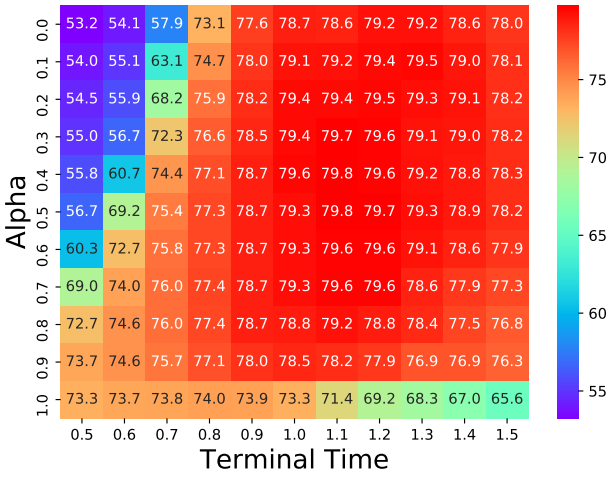

Figure 17: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Pubmed dataset.

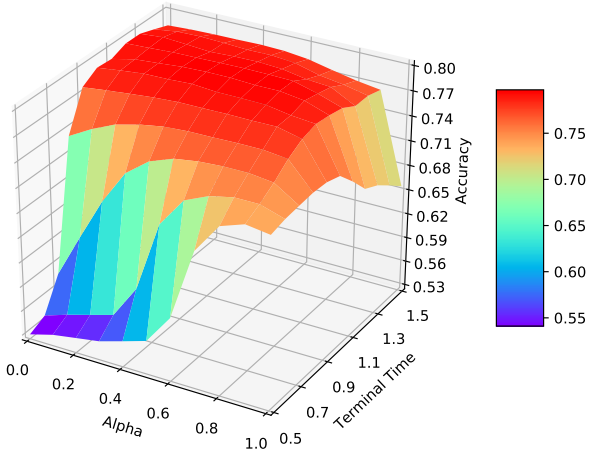

Figure 18: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Pubmed dataset in 3D surface plot.