Neural Dynamics on Complex Networks

Chengxi Zang
Department of Population Health Sciences, Weill Cornell Medicine
chz4001@med.cornell.edu

Fei Wang
Department of Population Health Sciences, Weill Cornell Medicine
few2001@med.cornell.edu

ABSTRACT

Learning continuous-time dynamics on complex networks is crucial for understanding, predicting and controlling complex systems in science and engineering. However, this task is very challenging due to the combinatorial complexities in the structures of high dimensional systems, their elusive continuous-time nonlinear dynamics, and their structural-dynamic dependencies. To address these challenges, we propose to combine Ordinary Differential Equation Systems (ODEs) and Graph Neural Networks (GNNs) to learn continuous-time dynamics on complex networks in a data-driven manner. We model differential equation systems by GNNs. Instead of mapping through a discrete number of neural layers in the forward process, we integrate GNN layers over continuous time numerically, leading to capturing continuous-time dynamics on graphs. Our model can be interpreted as a Continuous-time GNN model or a Graph Neural ODEs model. Our model can be utilized for continuous-time network dynamics prediction, structured sequence prediction (a regularly-sampled case), and node semi-supervised classification tasks (a one-snapshot case) in a unified framework. We validate our model by extensive experiments in the above three scenarios. The promising experimental results demonstrate our model's capability of jointly capturing the structure and dynamics of complex systems in a unified framework.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; **Ordinary differential equations**; • **Computing methodologies → Neural networks**; **Network science**;

KEYWORDS

Continuous-time Graph Neural Networks; Graph Neural Ordinary Differential Equations; Continuous-time GNNs; Graph Neural ODEs; Continuous-time Network Dynamics Prediction; Structured Sequence Prediction; Graph Semi-supervised Learning; Differential Deep Learning on Graphs;

ACM Reference Format:
Chengxi Zang and Fei Wang. 2020. Neural Dynamics on Complex Networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403132

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403132

1 INTRODUCTION

Real-world complex systems, such as brain [14], ecological systems [13], gene regulation [2], human health [5], and social networks [21, 24, 46–49], etc., are usually modeled as complex networks and their evolution are governed by some underlying nonlinear dynamics [30]. Revealing these dynamics on complex networks modeled by differential equation systems 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 [38],statistical physics [30], engineering [37], ecology [13] and biology [5], 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 dynamics on complex networks.

The development of a successful data-driven approach for modeling continuous-time dynamics on complex networks is very challenging: the interaction structures of the nodes in real-world networks are complex and the number of nodes and edges is large, which is referred to as the high-dimensionality of complex systems; the rules governing the dynamic change of nodes' states in complex networks are continuous-time and nonlinear; the structural and dynamic dependencies within the system are difficult to model by explicit mathematical functions. Recently, there has been an emerging trend in the data-driven discovery of ordinary differential equations (ODE) or partial differential equations (PDE) to capture the continuous-time dynamics, including sparse regression method [27, 34], residual network [31], feedforward neural network [33], etc. However, these methods can only handle very small ODE or PDE systems which consist of only a few interaction terms. For example, the sparse-regression-based method [27] shows that its combinatorial complexity grows with the number of agents when building candidate interaction library. Effective learning of continuous-time dynamics on large networks which consist of tens of thousands of agents and interactions is still largely unknown.

In this paper, we propose to combine Ordinary Differential Equation Systems (ODEs) and Graph Neural Networks (GNNs) [43] to learn non-linear, high-dimensional and continuous-time dynamics on graphs [1]. We model differential equation systems by GNNs to

[1]We use graph to avoid the ambiguity with neural network. Otherwise, we use graph and network interchangeably to refer to linked objects.

capture the instantaneous rate of change of nodes' states. Instead of mapping through a discrete number of layers in the forward process of conventional neural network models [20], we *integrate* [8] GNN layers over continuous time rather than discrete depths, leading to a novel *Continuous-time GNN* model. In a dynamical system view, the continuous depth can be interpreted as continuous physical time, and the outputs of any hidden GNN layer at time t are instantaneous network dynamics at that moment. Thus we can also interpret our model as a *Graph Neural ODEs* in analogy to the Neural ODEs [8]. 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**).

Our NDCN model can be used for following three tasks in *a unified framework*: 1) (Continuous-time network dynamics prediction): Can we predict the continuous-time dynamics on complex networks at an arbitrary time? 2) (Structured sequence prediction [36]): Can we predict next few steps of structured sequences? 3) (Graph semi-supervised classification [19, 44]): Can we infer the labels of each node given features for each node and some labels at one snapshot? The experimental results show that our NDCN can successfully finish above three tasks. As for the task 1, our NDCN is first of its kind which learns continuous-time dynamics on large complex networks modeled by differential equations systems. As for the task 2, our NDCN achieves lower error in predicting future steps of the structured sequence with much fewer parameters than the temporal graph neural network models [17, 28, 36, 45]. As for the task 3, our NDCN learns a continuous-time dynamics to spread features and labels of nodes to predict unknown labels of nodes, showing very competitive performance compared to many graph neural networks [19, 39, 40]. Our framework potentially serves as a unified framework to jointly capture the structure and dynamics of complex systems in a data-driven manner.

It's worthwhile to summarize our contributions as follows:

- **A novel model:** We propose a novel model NDCN, which combines Ordinary Differential Equation Systems (ODEs) and Graph Neural Networks (GNNs) to learn continuous-time dynamics on graphs. Such a model can tackle continuous-time network dynamics prediction, structured sequence prediction, and node semi-supervised classification on graphs in a unified framework.
- **Physical interpretations:** Instead of mapping neural networks through a discrete number of layers, we integrate differential equation systems modeled by GNN over continuous time, which gives physical meaning of continuous-time dynamics on graphs to GNNs. Our NDCN can be interpreted as a Continuous-time GNN model or a Graph Neural ODEs.
- **Good performance:** Our experimental results show that our NDCN learns continuous-time dynamics on various complex networks accurately, achieves lower error in structured sequence prediction with much fewer parameters than temporal graph neural network models, and outperforms many GNN models in node semi-supervised classification tasks.

Our codes and datasets are open-sourced at https://github.com/calvin-zcx/ndcn .

2 RELATED WORK

Dynamics of complex networks. Real-world complex systems are usually modeled as complex networks and driven by continuous-time nonlinear dynamics: the dynamics of brain and human microbial are examined in [14] and [5] respectively; [13] investigated the resilience dynamics of complex systems. [4] gave a pipeline to construct network dynamics. To the best of our knowledge, our NDCN is the first neural network approach which learns continuous-time dynamics on complex networks in a data-driven manner.

Data-driven discovery of differential equations. Recently, some data-driven approaches are proposed to learn the ordinary/-partial differential equations (ODEs or PDEs), including sparse regression [27], residual network [31], feedforward neural network [33], coupled neural networks [32] and so on. [27] tries to learn continuous-time biological networks dynamics by sparse regression over a large library of interaction terms. Building the interaction terms are prohibitively for large systems with many interactions due to its combinatorial complexity. 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 and Optimal control. Inspired by residual network [16] and ordinary differential equation (ODE) theory [25, 35], seminal work neural ODE model [8] was proposed to re-write residual networks, normalizing flows, and recurrent neural network in a dynamical system way. See improved Neural ODEs in [10]. However, our NDCN model deals with large differential equations systems. Besides, our model solves different problems, namely learning continuous-time dynamics on graphs. Relationships between back-propagation in deep learning and optimal control theory are investigated in [6, 15]. 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 graph neural differential equations systems. Our model solves novel tasks, e.g. learning the dynamics on complex networks and refer to Sec.3.1.

Graph neural networks and temporal-GNNs. Graph neural networks (GNNs) [43], e.g., Graph convolution network (GCN) [19], attention-based GNN (AGNN) [39], graph attention networks (GAT) [40], etc., achieved very good performance on node semi-supervised learning on graphs. However, existing GNNs usually have integer number of 1 or 2 layers [22, 43]. Our NDCN gives a dynamical system view to GNNs: the continuous depth can be interpreted as continuous physical time, and the outputs of any hidden layer at time t are instantaneous rate of change of network dynamics at that moment. By capturing continuous-time network dynamics with real number of depth/time, our model gives very competitive and even better results than above GNNs. By combining RNNs or convolution operators with GNNs, temporal-GNNs [17, 28, 36, 45] try to predict next few steps of 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, but also predicts the structured sequences very well with much fewer parameters.

3 GENERAL FRAMEWORK

3.1 Problem Definition

We can describe the continuous-time dynamics on a graph by a differential equation system:

$$\frac{dX(t)}{dt} = f\Big(X, G, W, t\Big), \tag{1}$$

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 nodes are interacted with 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 $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ governs the instantaneous rate of change of dynamics on the graph. In addition, nodes can have various semantic labels encoded by one-hot encoding $Y(X, \Theta) \in \{0, 1\}^{n\times k}$, and Θ represents the parameters of this classification function. The problems we are trying to solve are:

- **(Continuous-time network dynamics prediction) How to predict the continuous-time dynamics $\frac{dX(t)}{dt}$ on a graph at an arbitrary time?** Mathematically, given a graph G and nodes' states of system $\{X(\hat{t}_1), X(\hat{t}_2), ..., X(\hat{t}_T)|0 \le t_1 < ... < t_T\}$ where t_1 to t_T are arbitrary physical timestamps, can we learn differential equation systems $\frac{dX(t)}{dt} = f(X, G, W, t)$ to predict continuous-time dynamics $X(t)$ at an arbitrary physical time t? The arbitrary physical times mean that $\{t_1, ..., t_T\}$ are irregularly sampled with different observational time intervals. When $t > t_T$, we call the prediction task **extrapolation prediction**, while $t < t_T$ and $t \neq \{t_1, ..., t_T\}$ for **interpolation prediction**.
- **(Structured sequence prediction)**. As a special case when $X(t)$ are sampled regularly with the same time intervals $\{X\hat{[}1], X\hat{[}2], ..., X\hat{[}T]\}$, the above problem degenerates to a structured sequence learning task with an emphasis on sequential order instead of arbitrary physical times. The goal is to predict next m steps' structured sequence $X[T+1], ..., X[T+m]$.
- **(Node semi-supervised classification on graphs) How to predict the unknown nodes' labels given features for each node and some labels at one snapshot?** As a special case of above problem with an emphasis on a specific moment, given a graph G, one-snapshot features X and some labels $Mask \odot Y$, can we learn a network dynamics to predict unknown labels $(1 - Mask) \odot Y$ by spreading given features and labels on the graph?

We try to solve above three tasks on learning dynamics on graphs in a unified framework.

3.2 A Unified Learning Framework

We formulate our basic framework as follows:

$$\begin{aligned} \underset{W(t),\Theta(T)}{\arg\min}\ & \mathcal{L} = \int_0^T \mathcal{R}\Big(X, G, W, t\Big)\, dt + \mathcal{S}\Big(Y(X(T), \Theta)\Big) \\ \text{subject to}\ & \frac{dX(t)}{dt} = f\Big(X, G, W, t\Big), \quad X(0) = X_0 \end{aligned} \tag{2}$$

where $\int_0^T \mathcal{R}\Big(X, G, W, t\Big)\, dt$ is the "running" loss of the continuous-time dynamics on graph from $t = 0$ to $t = T$, and $\mathcal{S}(Y(X(T), \Theta))$ is the "terminal" loss at time T. By integrating $\frac{dX}{dt} = f(X, G, W, t)$ over time t from initial state X_0, a.k.a. solving the initial value problem [7] for this differential equation system, we can get the continuous-time network dynamics $X(t) = X(0) + \int_0^T f(X, G, W, \tau)\, d\tau$ at arbitrary time moment $t > 0$.

Such a formulation can be seen as an optimal control problem so that the goal becomes to learn the best control parameters $W(t)$ for differential equation system $\frac{dX}{dt} = f(X, G, W, t)$ and the best classification parameters Θ for semantic function $Y(X(t), \Theta)$ by solving above optimization problem. Different from traditional optimal control framework, we model the differential equation systems $\frac{dX}{dt} = f(X, G, W, t)$ by graph neural networks. By integrating $\frac{dX}{dt} = f(X, G, W, t)$ over continuous time, namely $X(t) = X(0) + \int_0^t f\Big(X, G, W, \tau\Big)\, d\tau$, we get our *graph neural ODE* model. In a dynamical system view, our graph neural ODE can be a time-varying coefficient dynamical system when $W(t)$ changes over time; or a constant coefficient dynamical system when W is constant over time for parameter sharing. It's worthwhile to recall that the deep learning methods with L hidden neural layers f_* are $X[L] = f_L \circ ... \circ f_2 \circ f_1(X[0])$, which are iterated maps [38] with an integer number of discrete layers and thus can not learn continuous-time dynamics $X(t)$ at arbitrary time. In contrast, our graph neural ODE model $X(t) = X(0) + \int_0^t f\Big(X, G, W, \tau\Big)\, d\tau$ can have continuous layers with a real number t depth corresponding to the continuous-time dynamics on graph G. Thus, we can also interpret our graph neural ODE model as a *continuous-time GNN*.

Moreover, to further increase the express ability of our model, we encode the network signal $X(t)$ from the original space to $X_h(t)$ in a hidden space, and learn the dynamics in such a hidden space. Then our model becomes:

$$\begin{aligned} \underset{W(t),\Theta(T)}{\arg\min}\ & \mathcal{L} = \int_0^T \mathcal{R}\Big(X, G, W, t\Big)\, dt + \mathcal{S}\Big(Y(X(T), \Theta)\Big) \\ \text{subject to}\ & X_h(t) = f_e\Big(X(t), W_e\Big), \quad X(0) = X_0 \\ & \frac{dX_h(t)}{dt} = f\Big(X_h, G, W_h, t\Big), \\ & X(t) = f_d\Big(X_h(t), W_d\Big) \end{aligned} \tag{3}$$

where 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 deep neural structures*. We name our graph neural ODE (or continuous-time GNN) model as *Neural Dynamics on Complex Networks* (**NDCN**).

We solve the initial value problem (i.e., integrating the differential equation systems over time numerically) by numerical methods (e.g., 1^{st}-order Euler method, high-order method Dormand-Prince DOPRI5 [9], etc.). The numerical methods can approximate continuous-time dynamics $X(t) = X(0) + \int_0^t f\Big(X, G, W, \tau\Big)\, d\tau$ at arbitrary time t accurately with guaranteed error. Thus, an equivalent formulation of Eq.(3) by explicitly solving the initial value

problem of differential equation system is:

$$\begin{aligned} \underset{W(t),\Theta(T)}{\arg\min} \quad & \mathcal{L} = \int_0^T \mathcal{R}\Big(X,\, G,\, W,\, t\Big)\, dt + \mathcal{S}\Big(Y(X(T),\Theta)\Big) \\ \text{subject to} \quad & X_h(t) = f_e\Big(X(t),\, W_e\Big), \quad X(0) = X_0 \\ & X_h(t) = X_h(0) + \int_0^t f\Big(X_h,\, G,\, W_h,\, \tau\Big) d\tau \\ & X(t) = f_d\Big(X_h(t),\, W_d\Big) \end{aligned} \tag{4}$$

A large time t corresponds to "deep" neural networks with the physical meaning of a long trajectory of dynamics on graphs. 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 [18]). We will show concrete examples of above general framework in the following three sections.

4 LEARNING CONTINUOUS-TIME NETWORK DYNAMICS

In this section, we investigate if our NDCN model can learn continuous-time dynamics on graphs for both interpolation prediction and extrapolation prediction.

4.1 A Model Instance

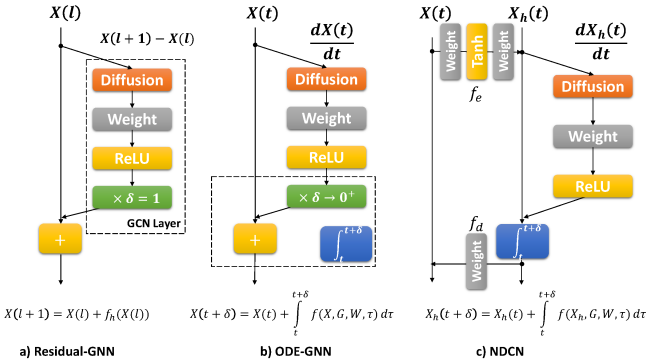

Figure 1: *Illustration of an NDCN instance.* a) a Residual Graph Neural Network (Residual-GNN), b) an Ordinary Differential Equation GNN model (ODE-GNN), and c) An instance of our Neural Dynamics on Complex Network (NDCN) model. The integer l represents the discrete l^{th} layer and the real number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

We solve the objective function in (3) with an emphasis on running loss and a fixed graph. 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 [19] but with a simplified diffusion operator Φ to model the instantaneous rate of change of dynamics on graphs in the hidden space, and a linear decoding function f_d for regression tasks in the original signal space. Thus, our model is:

$$\begin{aligned} \underset{W_*,\, b_*}{\arg\min} \quad & \mathcal{L} = \int_0^T |X(t) - \hat{X(t)}|\, dt \\ \text{subject to} \quad & X_h(t) = \tanh\Big(X(t)W_e + b_e\Big)W_0 + b_0, \quad X_0 \\ & \frac{dX_h(t)}{dt} = \text{ReLU}\Big(\Phi X_h(t) W + b\Big), \\ & X(t) = X_h(t)W_d + b_d \end{aligned} \tag{5}$$

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 illustrate the neural structures of our model in Figure 1c.

We adopt a linear 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. We design the graph neural differential equation system as $\frac{dX(t)}{dt} = \text{ReLU}(\Phi X(t)W + b)$ to learn any unknown network dynamics. We can regard $\frac{dX(t)}{dt}$ as a single neural layer at time moment t. The $X(t)$ at arbitrary time t is achieved by integrating $\frac{dX(t)}{dt}$ over time, i.e., $X(t) = X(0) + \int_0^t \text{ReLU}\Big(\Phi X(\tau)W + b\Big)\, d\tau$, leading to a continuous-time graph neural network.

4.2 Experiments

Network Dynamics We investigate following three continuous-time network dynamics from physics and biology. Let $\overrightarrow{x_i(t)} \in \mathbb{R}^{d\times 1}$ be d dimensional features of node i at time t and thus $X(t) = [\ldots, \overrightarrow{x_i(t)}, \ldots]^T \in \mathbb{R}^{n\times d}$. We show their differential equation systems in vector form for clarity and implement them in matrix form:

- 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 [26], 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 [13] capture the abundance $\overrightarrow{x_i(t)}$ of species i, consisting of incoming migration term b_i, logistic growth with population capacity k_i [47] and Allee effect [1] 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 [2, 13].

Complex Networks. We consider following networks: (a) Grid network, where each node is connected with 8 neighbors (as shown in Fig. 2(a)) ; (b) Random network, generated by Erdós and Rényi model [11] (as shown in Fig. 2(b)); (c) Power-law network, generated by Albert-Barabási model [3] (as shown in Fig. 2(c)); (d) Small-world network, generated by Watts-Strogatz model [41] (as shown in Fig. 2(d)); and (e) Community network, generated by random partition model [12] (as shown in Fig. 2(e)).

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 [29] 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 A for the animations of these dynamics on different complex networks over time.

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 same and construct 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 ordinary differential equation GNN model (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 neural ODE model [8], 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, namely weight layer 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 (See Fig. 1c without a weight layer).

Experimental setup. We generate underlying networks with 400 nodes by network models in Sec.4.2 and the illustrations are shown in Fig. 2,3 and 4. 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 A).

We irregularly sample 120 snapshots of the continuous-time dynamics $\{X(\hat{t}_1), ..., X(\hat{t}_{120})|0 \le t_1 < ... < t_{120} \le T\}$ where the time intervals between $t_1, ..., t_{120}$ are different. We randomly choose 80 snapshots from $X(\hat{t}_1)$ to $X(\hat{t}_{100})$ for training, the left 20 snapshots from $X(\hat{t}_1)$ to $X(\hat{t}_{100})$ for testing the interpolation prediction task. We use the 20 snapshots from $X(\hat{t}_{101})$ to $X(\hat{t}_{120})$ for testing the extrapolation prediction task.

We use Dormand-Prince method [9] to get the ground truth dynamics, and use Euler method in the forward process of our NDCN (More configurations in Appendix B). We evaluate the results by ℓ_1 loss and normalized ℓ_1 loss (normalized by the mean element-wise value of $X(\hat{t})$), and they lead to the same conclusion (We report normalized ℓ_1 loss here and see Appendix C for ℓ_1 loss). Results are the mean and standard deviation of the loss over 20 independent runs for 3 dynamic laws on 5 different networks by each method.

Table 1: Extrapolation of continuous-time network dynamics. 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	**4.1 ± 1.2**	**4.3 ± 1.6**	**4.9 ± 0.5**	**2.5 ± 0.4**	**4.8 ± 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	**26.7 ± 4.7**	**3.8 ± 1.8**	**7.4 ± 2.6**	**14.4 ± 3.3**	**3.6 ± 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	**16.0 ± 7.2**	**1.8 ± 0.5**	**3.6 ± 0.9**	**4.3 ± 0.9**	**2.5 ± 0.6**

Table 2: Interpolation of continuous-time network dynamics. 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	**3.2 ± 0.6**	**3.2 ± 0.4**	**5.6 ± 0.6**	**3.4 ± 0.4**	**4.3 ± 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	**7.6 ± 1.1**	**6.6 ± 2.4**	**6.5 ± 1.3**	**4.7 ± 0.7**	**7.9 ± 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	**5.8 ± 1.0**	**1.5 ± 0.6**	**2.9 ± 0.5**	**4.2 ± 0.9**	**2.3 ± 0.6**

Results. We visualize the ground-truth and learned dynamics in Fig. 2,3 and 4, and please see the animations of these network dynamics in Appendix A. 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. 4a but unstable on random networks in Fig. 4b or community networks in Fig. 4e. Both gene regulation dynamics in Fig. 4c and biological mutualistic dynamics in Fig. 3c show very bursty patterns on power-law networks. However, visually speaking, our NDCN learns all these different network dynamics very well.

The quantitative results of extrapolation and interpolation prediction are summarized in Table 1 and Table 2 respectively. 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.

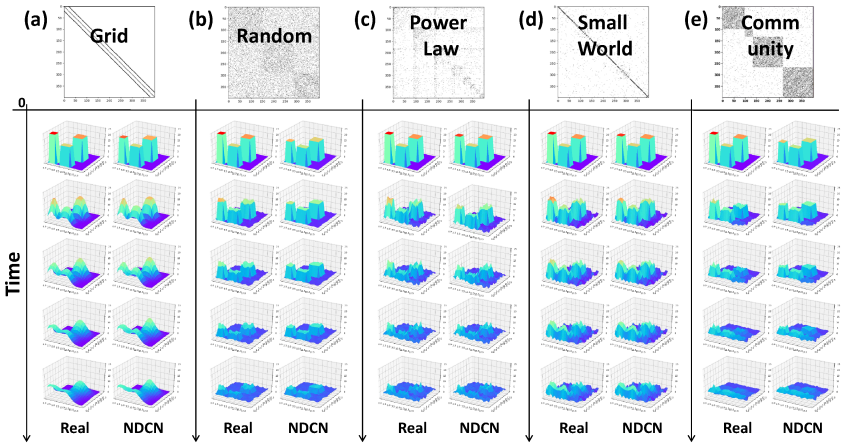

Figure 2: ***Heat diffusion on different networks. Our NDCN fits the dynamics on different networks accurately.*** **Each of the five vertical panels (a)-(e) represents the dynamics on one network over physical time. For each network dynamics, we illustrate the sampled ground truth dynamics (left) and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

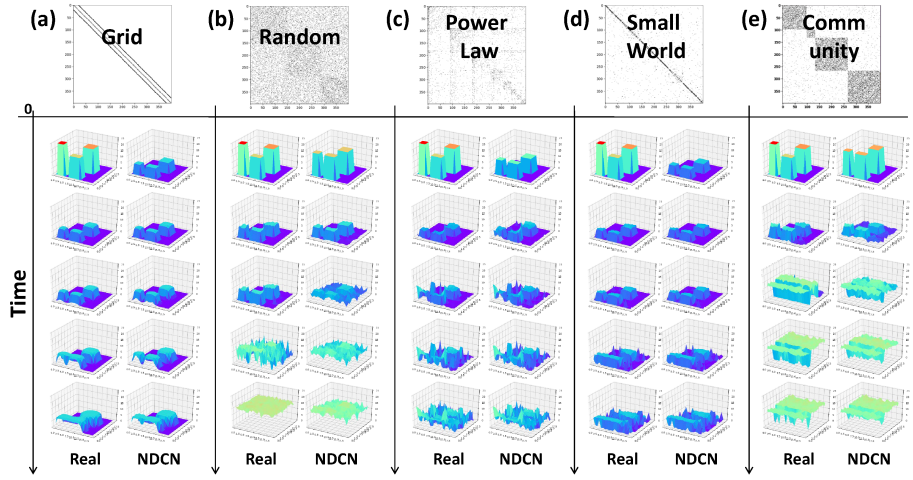

Figure 3: ***Biological mutualistic interaction on different networks. Our NDCN fits the dynamics on different networks accurately. Refer to Appendix A for the animations.***

5 LEARNING STRUCTURED SEQUENCES

Besides, we can easily use our NDCN model for learning regularly-sampled structured sequence of network dynamics and predict future steps by using 1^{st}-order Euler method with time step 1 in the forward integration process.

Experimental setup. We regularly sample 100 snapshots of the continuous-time network dynamics discussed in the last section with same time intervals from 0 to T, and denote these structured sequence as $\{X\hat{[}1], ..., X[\hat{100}]\}$. We use first 80 snapshots $X\hat{[}1], ..., X[\hat{80}]$ for training and the left 20 snapshots $X[\hat{81}], ..., X[\hat{100}]$ for testing extrapolation prediction task. The temporal-GNN models are usually used for next few step prediction and can not be used for the interpolation task (say, to predict $X[1.23]$) directly. We use 5 and 10 for hidden dimension of GCN and RNN models respectively. We use 1^{st}-order Euler method with time step 1 in the forward integration process. Other settings are the same as previous continuous-time dynamics experiment.

Baselines. We compare our model with the temporal-GNN models which are usually combinations of RNN models and GNN models [17, 28, 36]. We use GCN [19] as a graph structure extractor and use LSTM/GRU/RNN [23] to learn the temporal relationships between ordered structured sequences. We keep the loss function same and construct following baselines:

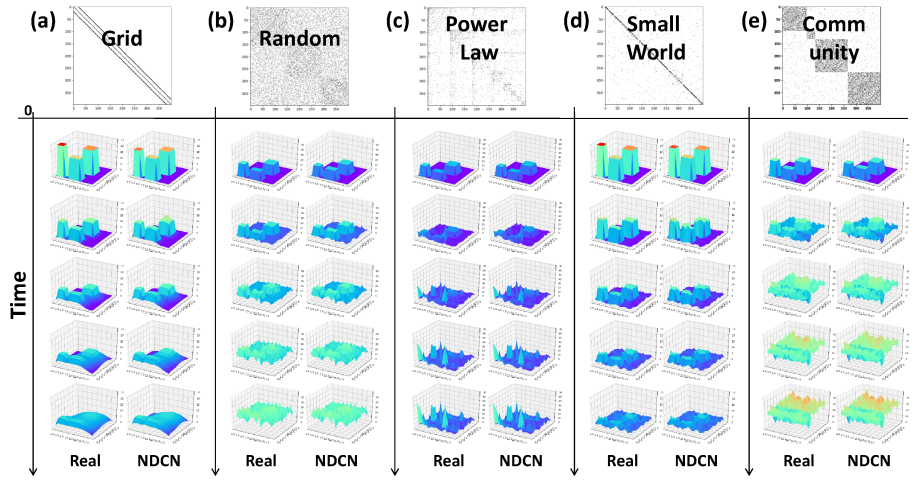

Figure 4: ***Gene regulation dynamics on different networks. Our NDCN fits the dynamics on different networks accurately. Refer to Appendix A for the animations.***

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

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

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

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

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

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

Results. We summarize the results of the extrapolation prediction of regularly-sampled structured sequence in Table 3. The GRU-GNN model works well in mutualistic dynamics on random network and community network. Our NDCN predicts different dynamics on these complex networks accurately and outperforms the baselines in almost all the settings. What's more, our model predicts the structured sequences in a much more succinct way with much fewer parameters. The learnable parameters of RNN-GNN, GRU-GNN, LSTM-GNN are 24530, 64770, and 84890 respectively. In contrast, our NDCN has only **901** parameters, accounting for 3.7%, 1.4% , 1.1% of above three temporal-GNN models respectively.

Table 3: Extrapolation prediction for the regularly-sampled structured sequence. 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	$\mathbf{18.6 \pm 9.9}$	$\mathbf{2.4 \pm 0.9}$	$\mathbf{4.1 \pm 1.4}$	$\mathbf{5.5 \pm 0.8}$	$\mathbf{2.9 \pm 0.5}$

6 NODE SEMI-SUPERVISED CLASSIFICATION

We investigate the third question, i.e., how to predict the semantic labels of each node given semi-supervised information? Various graph neural networks (GNN) [43] achieve very good performance in graph semi-supervised classification task [19, 44]. Existing GNNs usually adopt an integer number of 1 or 2 hidden layers [19, 40]. Our framework follows the perspective of a dynamical system: by modeling the continuous-time dynamics to spread nodes' features and given labels on graphs, we predict the unknown labels in analogy to predicting the label dynamics at some time T.

6.1 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:

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

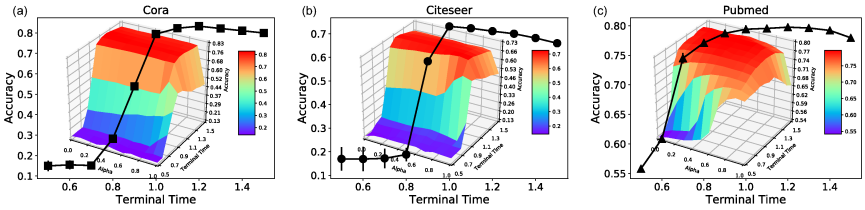

Figure 5: *Our NDCN model captures continuous-time dynamics.* Mean classification accuracy of 100 runs over terminal time when given a specific α. Insets are the accuracy over the two-dimensional space of terminal time and α. The best accuracy is achieved at a real-number time/depth.

Table 4: 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

where the $\hat{Y} \in \mathbb{R}^{n\times c}$ is the supervised information of nodes' labels and $Y(T) \in \mathbb{R}^{n\times c}$ is the label dynamics 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 $k = 1, \ldots, c$. We use differential equation system $\frac{dX(t)}{dt} = \mathrm{ReLU}(\Phi X(t))$ together with encoding and decoding layers to spread the features and given labels of nodes on graph over continuous time $[0, T]$, i.e., $X_h(T) = X_h(0) + \int_0^T \mathrm{ReLU}\left(\Phi X_h(t)\right) dt$.

Compared with the model in Eq.5, we only have a one-shot supervised information $\hat{Y}$ given nodes' features X. Thus, 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. 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, and when $\alpha = 0.5$, Φ averages both neighbors and itself as in [19].

6.2 Experiments

Datasets and Baselines. We use 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 [19, 39, 44]. We summarize the datasets in Table 4. We compare our NDCN model with graph convolution network (GCN) [19], attention-based graph neural network (AGNN) [39], and graph attention networks (GAT) [40] with sophisticated attention parameters.

Experimental setup. For the consistency of comparison with prior work, we follow the same experimental setup as [19, 39, 40]. 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 4. 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 [9]. We train our model for a maximum of 100 epochs using Adam [18] 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 for 100 runs in Table 5. 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).

Results. We summarize the results in Table 5. We find our NDCN outperforms many state-of-the-art GNN models. Results for the baselines are taken from [19, 39, 40, 42]. We report the mean and standard deviation of our results for 100 runs. We get our reported results in Table 5 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. We find best accuracy is achieved at a real-number time/depth.

Table 5: 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	**79.9 ± 0.1**
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
NDCN	**83.3 ± 0.6**	**73.1 ± 0.6**	**79.8 ± 0.4**

By capturing the continuous-time network dynamics to diffuse features and given labels on graphs, our NDCN gives better classification accuracy at terminal time $T \in \mathbb{R}^+$. Figure 5 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 D). We find for all the three datasets their accuracy curves follow rise and fall patterns around the best terminal time stamps which are real number. 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. The prior GNNs can only have an discrete number of layers which can not capture the continuous-time network dynamics accurately. In contrast, our NDCN captures continuous-time dynamics on graphs in a more fine-grained manner.

7 CONCLUSION

We propose to combine differential equation systems and graph neural networks to learn continuous-time dynamics on complex networks. Our NDCN gives the meaning of physical time and the continuous-time network dynamics to the depth and hidden outputs of GNNs respectively, predicts continuous-time dynamics on complex network and regularly-sampled structured sequence accurately, and outperforms many GNN models in the node semi-supervised classification task (a one-snapshot case). Our model potentially serves as a unified framework to capture the structure and dynamics of complex systems in a data-driven manner. For future work, we try to apply our model to other applications including molecular dynamics and urban traffics. Codes and datasets are open-sourced at https://github.com/calvin-zcx/ndcn.

ACKNOWLEDGEMENT

This work is supported by NSF 1716432, 1750326, ONR N00014-18-1-2585, Amazon Web Service (AWS) Machine Learning for Research Award and Google Faculty Research Award.

REFERENCES

[1] Warder Clyde Allee, Orlando Park, Alfred Edwards Emerson, Thomas Park, Karl Patterson Schmidt, et al. 1949. *Principles of animal ecology*. Technical Report. Saunders Company Philadelphia, Pennsylvania, USA.

[2] Uri Alon. 2006. *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC.

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

[4] Baruch Barzel, Yang-Yu Liu, and Albert-László Barabási. 2015. Constructing minimal models for complex system dynamics. *Nature communications* (2015).

[5] Amir Bashan, Travis E Gibson, Jonathan Friedman, Vincent J Carey, Scott T Weiss, Elizabeth L Hohmann, and Yang-Yu Liu. 2016. Universality of human microbial dynamics. *Nature* 534, 7606 (2016), 259.

[6] Martin Benning, Elena Celledoni, Matthias J Ehrhardt, Brynjulf Owren, and Carola-Bibiane Schönlieb. 2019. Deep learning as optimal control problems: models and numerical methods. *arXiv preprint arXiv:1904.05657* (2019).

[7] William E Boyce, Richard C DiPrima, and Douglas B Meade. 1992. *Elementary differential equations and boundary value problems*. Vol. 9. Wiley New York.

[8] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*. 6571–6583.

[9] John R Dormand. 1996. *Numerical methods for differential equations: a computational approach*. Vol. 3. CRC Press.

[10] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. 2019. Augmented neural odes. *arXiv preprint arXiv:1904.01681* (2019).

[11] P Erdos and A Renyi. 1959. On random graphs I. *Publ. Math. Debrecen* 6 (1959).

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

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

[14] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. 2014. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

[15] Jiequn Han, Qianxiao Li, et al. 2018. A mean-field optimal control formulation of deep learning. *arXiv preprint arXiv:1807.01083* (2018).

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[17] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2019. Relational Representation Learning for Dynamic (Knowledge) Graphs: A Survey. *arXiv preprint arXiv:1905.11485* (2019).

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015*.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR 2017*.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[21] Haoyang Li, Peng Cui, Chengxi Zang, Tianyang Zhang, Wenwu Zhu, and Yishi Lin. 2019. Fates of Microscopic Social Ecosystems: Keep Alive or Dead?. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 668–676.

[22] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[23] Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *preprint arXiv:1506.00019* (2015).

[24] Yunfei Lu, Linyun Yu, Tianyang Zhang, Chengxi Zang, Peng Cui, Chaoming Song, and Wenwu Zhu. 2018. Collective Human Behavior in Cascading System: Discovery, Modeling and Applications. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 297–306.

[25] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. 2017. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121* (2017).

[26] A v Luikov. 2012. *Analytical heat diffusion theory*. Elsevier.

[27] Niall M Mangan, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2016. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* 2, 1 (2016), 52–63.

[28] Apurva Narayan and Peter HOâĂŹN Roe. 2018. Learning graph dynamics using deep neural networks. *IFAC-PapersOnLine* 51, 2 (2018), 433–438.

[29] Mark Newman. 2010. *Networks: an introduction*. Oxford U. press.

[30] Mark Newman, Albert-Laszlo Barabasi, and Duncan J Watts. 2011. *The structure and dynamics of networks*. Vol. 12. Princeton University Press.

[31] Tong Qin, Kailiang Wu, and Dongbin Xiu. 2018. Data driven governing equations approximation using deep neural networks. *arXiv preprint arXiv:1811.05537* (2018).

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

[33] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2018. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236* (2018).

[34] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2017. Data-driven discovery of partial differential equations. *Science Advances* 3, 4 (2017), e1602614.

[35] Lars Ruthotto and Eldad Haber. 2018. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272* (2018).

[36] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*. 362–373.

[37] Jean-Jacques E Slotine, Weiping Li, et al. 1991. *Applied nonlinear control*. Vol. 199. Prentice hall Englewood Cliffs, NJ.

[38] Steven H Strogatz. 2018. *Nonlinear Dynamics and Chaos with Student Solutions Manual: With Applications to Physics, Biology, Chemistry, and Engineering*.

[39] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735* (2018).

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

[41] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of âĂŸsmall-worldâĂŹnetworks. *nature* 393, 6684 (1998), 440.

[42] Felix Wu, Tianyi Zhang, Amauri H. Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. *CoRR* (2019).

[43] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[44] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML 2016*. 40–48.

[45] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[46] Chengxi Zang, Peng Cui, and Christos Faloutsos. 2016. Beyond sigmoids: The nettide model for social network growth, and its applications. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2015–2024.

[47] Chengxi Zang, Peng Cui, Christos Faloutsos, and Wenwu Zhu. 2018. On Power Law Growth of Social Networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1727–1740.

[48] Chengxi Zang, Peng Cui, Chaoming Song, Wenwu Zhu, and Fei Wang. 2019. Uncovering Pattern Formation of Information Flow. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1691–1699.

[49] Chengxi Zang, Peng Cui, Wenwu Zhu, and Fei Wang. 2019. Dynamical Origins of Distribution Functions. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 469–478.

APPENDIX:

A 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.

B MODEL CONFIGURATIONS

Model configurations of learning network dynamics in both continuous-time and regularly-sampled settings. We train our NDCN model by Adam [18]. 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 [18] with learning rate 0.01. We summarize our ℓ_2 regularization parameter as in Table 6 and Table 7 for Section 4 learning continuous-time network dynamics. We summarize our ℓ_2 regularization parameter as in Table 8 for Section 5 learning regularly-sampled dynamics.

Table 6: 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

Table 7: 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

Table 8: 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

C RESULTS IN ABSOLUTE ERROR.

We show corresponding ℓ_1 loss error in Table 9,Table 10 and Table 11 with respect to the normalized ℓ_1 loss error in Section 4 learning continuous-time network dynamics and Section 5 learning regularly-sampled dynamics. The same conclusions can be made as in Table 1,Table 2 and Table 3.

Table 9: 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	**0.158 ± 0.047**	**0.163 ± 0.060**	**0.187 ± 0.020**	**0.097 ± 0.016**	**0.183 ± 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	**1.038 ± 0.181**	**0.584 ± 0.277**	**0.653 ± 0.230**	**0.521 ± 0.124**	**0.502 ± 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	**1.089 ± 0.487**	**0.715 ± 0.210**	**0.342 ± 0.088**	**0.243 ± 0.051**	**0.782 ± 0.199**

Table 10: 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	**0.121 ± 0.024**	**0.121 ± 0.017**	**0.214 ± 0.024**	**0.129 ± 0.017**	**0.165 ± 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	**0.164 ± 0.031**	**0.843 ± 0.267**	**0.333 ± 0.055**	**0.085 ± 0.014**	**0.852 ± 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	**0.262 ± 0.046**	**0.455 ± 0.174**	**0.222 ± 0.034**	**0.180 ± 0.032**	**0.562 ± 0.130**

Table 11: 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	**0.165 ± 0.027**	**0.180 ± 0.063**	**0.208 ± 0.015**	**0.103 ± 0.014**	**0.201 ± 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	**0.162 ± 0.564**	1.077 ± 0.071	1.792 ± 0.165	**0.510 ± 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	**1.414 ± 0.060**	**0.734 ± 0.168**	**0.990 ± 0.442**	**0.557 ± 0.078**	**0.528 ± 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	**1.267 ± 0.672**	**0.946 ± 0.357**	**0.397 ± 0.133**	**0.312 ± 0.043**	**0.901 ± 0.160**

D 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.

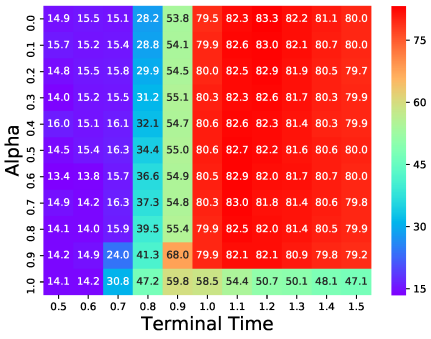

Figure 6: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Cora dataset in heatmap plot.

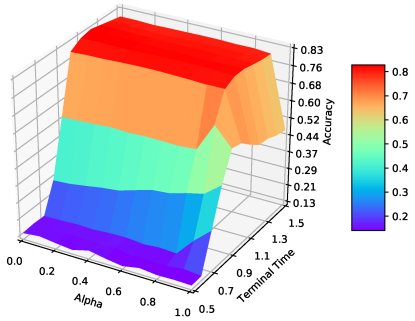

Figure 7: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Cora dataset in 3D surface plot.

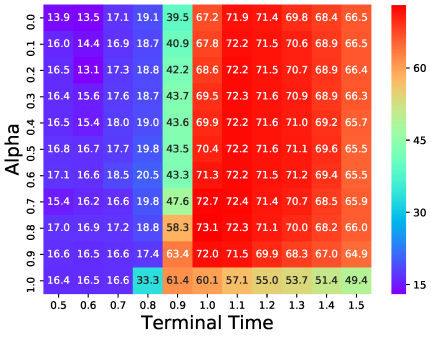

Figure 8: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Citeseer dataset.

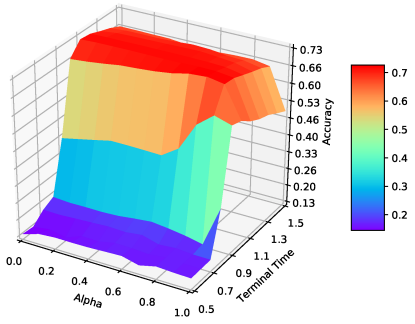

Figure 9: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Citeseer dataset in 3D surface plot.

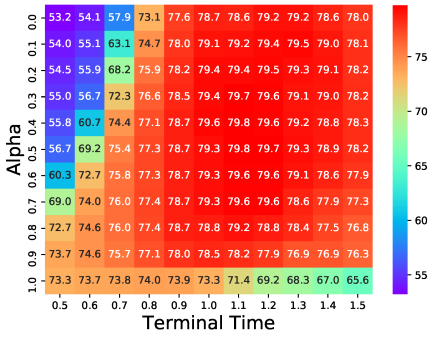

Figure 10: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Pubmed dataset.

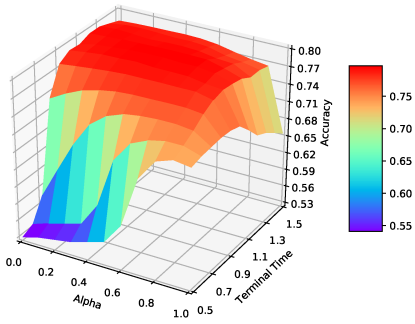

Figure 11: Mean classification accuracy of 100 runs of our NDCN model over terminal time and α for the Pubmed dataset in 3D surface plot.