EasyChair Preprint

№ 11110

AGGDN: a Continuous Stochastic Predictive Model for Monitoring Sporadic Time Series on Graphs

Yucheng Xing, Jacqueline Wu, Yingru Liu, Xuewen Yang and Xin Wang

October 23, 2023

AGGDN: A Continuous Stochastic Predictive Model for Monitoring Sporadic Time Series on Graphs

Yucheng Xing[1 [0000−0002−3052−6527]], Jacqueline Wu[2], Yingru Liu[1], Xuewen Yang[1,3], and Xin Wang[1 [0000−0001−8639−3818]]

[1] Stony Brook University, Stony Brook NY 11790, USA
[2] New York University, New York NY 10012, USA
[3] InnoPeak Technology, Inc., Palo Alto CA 94303, USA

Abstract. Monitoring data of real-world networked systems could be sparse and irregular due to node failures or packet loss, which makes it a challenge to model the continuous dynamics of system states. Representing a network as graph, we propose a deep learning model, *Adversarial Graph-Gated Differential Network (AGGDN)*. To accurately capture the spatial-temporal interactions and extract hidden features from data, AGGDN introduces a novel module, *dynDC-ODE*, which empowers Ordinary Differential Equation (ODE) with learning-based Diffusion Convolution (DC) to effectively infer relations among nodes and parameterize continuous-time system dynamics over graph. It further incorporates a Stochastic Differential Equation (SDE) module and applies it over graph to efficiently capture the underlying uncertainty of the networked systems. Different from any single differential equation model, the ODE part also works as a control signal to modulate the SDE propagation. With the recurrent running of the two modules, AGGDN can serve as an accurate online predictive model that is effective for either monitoring or analyzing the real-world networked objects. In addition, we introduce a soft masking scheme to capture the effects of partial observations caused by the random missing of data from nodes. As training a model with SDE component could be challenging, Wasserstein adversarial training is exploited to fit the complicated distribution. Extensive results demonstrate that AGGDN significantly outperforms existing methods for online prediction.

Keywords: Graph Sequence Prediction · Sporadic Time Series · Continuous Model · Stochastic Model

1 Introduction

Many systems, such as social networks, vehicle networks, communication networks, and power grids, are networked and can be represented as graphs. Although a practical system operates continuously, its states are normally collected periodically at discrete time instants. Due to practical constraints such as cost,

unreliable communications, device malfunction or failure, the observations of systems are often sporadic, which are sparse and irregular in both spatial and temporal domains. In order to timely and effectively control the systems for reliable and intelligent operations, it is important to accurately predict the system states based on the collected observations. The aim of this paper is to attack the challenge of modeling the graph dynamics where signals of nodes evolve continuously but the observations are sporadic.

It is highly non-trivial to learn the structured dynamics from sporadic observations on a graph. The challenges mainly come from three sources. First, the signals from nodes are time varying, and it is hard to model the spatial-temporal interaction across the whole graph. Second, only partial dynamics can be observed from sporadic data, which makes it difficult to model the underlying process. Last, as the output signals of networked systems may contain both process uncertainty (e.g. the distributed energy resource control signal of a microgrid is influenced by the uncertainty of its input.) and measurement noise, the distribution of data can be complicated and difficult to estimate.

With the rapid development of Graph Neural Network (GNN) [44, 38, 17], there are considerable number of studies on learning the dynamics of time series on graphs [16, 48, 46]. However, existing efforts mostly consider discrete-time dynamics, assuming systems are fully observed with complete data taken periodically. Although data missing is considered in [39, 4] for graph prediction, the scheme still considers discrete data samples and can not depict the ground-true dynamics of the continuous-time network systems in real world. Recently, several ODE-based models are proposed to learn the continuous dynamics of the graph time series [35, 31, 12, 8, 2], but ODE is only applied to learn the deterministic dynamics. Neglecting the process uncertainty of the system, they are unable to capture the complicated distribution of observations in real-world systems.

We propose a continuous-time graph-based recurrent neural network, *Adversarial Graph-Gated Differential Network (AGGDN)*, to capture the underlying dynamics on the graph structure from sporadic observations of node signals. AGGDN models the stochastic process of dynamic states of networked systems with a novel compounded infrastructure with ODE modulated SDE running continuously over a graph. On the one hand, SDE module can supplement the deterministic ODE module with a stochastic variation; on the other hand, ODE part can work as a control signal to modulate the SDE propagation. In addition, in order to address the challenge of learning the complete system states with incomplete data samples, we first propose a soft-masking scheme that can better extract disentangled hidden features of partial observations to well explore the spatial-temporal relation in the graph. Furthermore, to more accurately parameterize the ODE module, we enhance diffusion convolution [20] with the learning of impacts of nodes in the graph and call it as *dynamic Diffusion Convolution (dynDC)*. The SDE component is incorporated to efficiently capture the process uncertainty of the underlying system dynamics with a flexible tracking of data distribution. However, optimizing an SDE model is difficult, so we further exploit Wasserstein adversarial training to efficiently train AGGDN.

The contributions of this paper can be summarized as follows:

1. We propose a novel AGGDN architecture to accurately predict system states of a networked system, given sparse and irregular data samples and system uncertainty. It accurately models the continuous-time process over graph with a compounded ODE-SDE structure, where SDE tracks the stochastic variation of states to capture system uncertainties and ODE provides a modulation for the whole model propagation.
2. We design a dynDC-ODE module with ODE empowered with the dynamic diffusion convolution to learn different impacts of nodes, that our model can well adapt to the dynamic changes of graph.
3. We introduce soft-masking that AGGDN can better learn from the partial observations of a networked system to more effectively infer missing data.
4. We employ Wasserstein adversarial training for our continuous-time AGGDN model, rather than taking variational inference methods (assumed by most SDE models) that are inefficient for high-dimensional data.

The rest of this paper is organized as follows. The problem formulation and related works are introduced in Sec. 2. The detailed architectures and training method of our model are proposed in Sec. 3. Extensive experiments and analysis are given in Sec. 4 and conclusions are made in Sec. 5. The source code and data are available at https://github.com/SBU-YCX/AGGDN.

2 Background

2.1 Problem Formulation

Notations We denote a graph with time-varying signals at nodes by $\mathbb{G} = \{\mathcal{V}, \mathcal{E}, \{\mathcal{X}_{t_n}, \mathcal{M}_{t_n}, t_n\}_{n=0}^{N}\}$, where $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ is the set of nodes and $\mathcal{E} = \{(v_i, v_j)|v_i, v_j \in \mathcal{V}\}$ is the set of edges between nodes. The cardinalities $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of elements in $\mathcal{V}$ and $\mathcal{E}$. We further denote the 0-1 adjacent matrix of a graph as A. In a given network $\{\mathcal{V}, \mathcal{E}\}$, the dynamic states are described by a sequence of N frames. A frame contains a multi-variate signal $\mathcal{X}_{t_n} \in \mathbb{R}^{|\mathcal{V}|\times d}$ captured at the discrete time $t_n \in \mathbb{R}_+$, where d is the corresponding dimension of the signal. Since sensing or transmission problems in real-world systems often cause sample missing, in each frame, a mask $\mathcal{M}_{t_n} = \{0,1\}^{|\mathcal{V}|\times d}$ is used to indicate if there exists a signal in the corresponding dimension. Therefore, the actual observation sequence $\mathcal{O} = \{\mathcal{X}_{t_n} \odot \mathcal{M}_{t_n}\}_{n=0}^{N}$ fed into the model is a sporadic time series with irregular data in both temporal and spatial domains. $\mathcal{O}_{t_n} = \mathcal{X}_{t_n} \odot \mathcal{M}_{t_n}$ denotes the input data at time t_n, where $\odot$ represents the element-wise multiplication between two matrices.

Objective Given a collection of data $D = \{\mathbb{G}^{(k)}\}_{k=1}^{|D|}$, where $\mathbb{G}^{(k)}$ is a data sequence introduced in Notation part above and $|D|$ is the total number of such sequences in the dataset, our goal is to learn a continuous-time recurrent predictive model $\mathcal{G}$ to maximize the masked log-likelihood:

$$\mathcal{L}_{ll}(\mathcal{G}) = \mathbb{E}_{\mathbb{G}^{(k)} \in D} \sum_{n=1}^{N} \mathcal{M}_{t_n} \otimes \log P_{\mathcal{G}}(\mathcal{X}_{t_n} | \mathcal{O}_{t_0:t_{n-1}}, A), \tag{1}$$

where $\otimes$ is defined as the sum of element-wise product of two matrices, $P_{\mathcal{G}}(\cdot)$ denotes the probability density of each element in the feature matrices. We want to emphasize that the log-likelihood is only evaluated on the observed training data indicated by the binary masks instead of full data. Therefore, the objective described in (1) can be regarded as an unsupervised one.

2.2 Related Works

Graph Convolution It is the core operation of Graph Convolutional Network (GCN) [6, 17, 20]. To explore the relation among multi-hop neighbors in a graph, diffusion convolution was proposed in [20]. Rather than only using a binary adjacency matrix to indicate if there exist edges between nodes, we are interested in actively learning the relation among nodes for the more accurate modeling and thus more accurate prediction of dynamic system states. Our model, however, does not depend on the choice of graph convolutional operators and can be straight-forwardly adapted for other graph convolution methods.

Graph Recurrent Networks Most existing sequential graph models are proposed for discrete-time data [48, 36, 28, 47, 7, 29], and assume that the data sequence is fully observed. Recurrent models have also been applied to non-sequential data on static graphs, for applications such as graph generation [45, 22, 23, 37, 3] and feature learning [14, 21, 40, 13]. DynGEM [10] and dyngraph2vec [9] are deep learning models for tracking the structure evolution of the graph topology. Our study, however, focuses mainly on modeling the stochastic process of the signals on the graph [11, 43], especially under partial observations.

Neural Differential Equations (NeuralODE) NeuralODE [1] incorporates a deep learning module to parameterize nonlinear ordinary differential equation (ODE). In order to better model the continuous-time process of the vectorized data sequence, the structure of NeuralODE is further extended [34, 5] by introducing a recurrent component to efficiently integrate the data information into the feature trajectories. Some ODE-based studies [35, 31, 12, 8, 2] are made to learn the continuous-time dynamics of node features on a graph, assuming that the underlying dynamics are deterministic and neglecting the process uncertainty existing in many real-world systems. These models can merely parameterize a simplified data distribution, not to say capture the complicated stochastic process of the systems with both process and measurement uncertainties.

To better model the randomness of data, neural stochastic differential equation (NeuralSDE) is proposed to bridge the gap between nonlinear SDE and deep learning models [25, 30, 18, 42, 41, 19]. In [25, 30, 18], neural network components are introduced into SDE to define more robust and accurate deep learning architectures to solve supervised learning problems such as image classification.

A scalable method is proposed in [19] to compute the gradients for optimizing NeuralSDE. To the best of our knowledge, most NeuralSDE models [24, 26, 27, 19] are proposed for vector or matrix data but not for representing time series on graphs. Our focus, however, is on accurately predicting system states under dynamics and data missing through sequential learning over graphs.

We develop a compound model with the integration of NeuralODE and NeuralSDE into one infrastructure. It not only extends over a graph to capture the spatial interaction of data in multiple hops but also simultaneously track the deterministic dynamics and process uncertainty through our proposed dynamic Diffusion Convolution (dynDC). In a single ODE or SDE model, the propagation within an interval between two time stamps only relies on the preceding observed values and make an update at any time point directly. Instead, our model first gets an approximate prediction through the ODE part, which is then used as a control signal to modulate the latent state thus the final data predicted have more continuous changes. The natural cubic spline interpolation method [15, 2] makes a three-degree polynomial assumption about the curve between two observations, but needs a full time series to construct the underlying process approximation. Instead, ODE module allows creating the corresponding control signal as time evolves. We further introduce a learning-based soft-masking scheme to work with our dynDC-ODE components that AGGDN can adapt to spatially irregular observations for a more accurate modeling and apply the Wasserstein adversarial training to efficiently train our model.

3 Methodology

We propose a flexible continuous-time recurrent model, Adversarial Graph-Gated Differential Network (AGGDN), which is capable of learning the continuous graph dynamics from spatial-temporal irregular observation sequences. We first introduce its architecture, then provide the details of the key components. To effectively train the stochastic part, we adopt the adversarial training strategy and will discuss the training details at the end of this section.

3.1 Architecture

As shown in Fig. 1, our model consists of two trajectories: one hidden feature trajectory to integrate the topological relation in the graph and one latent state trajectory to capture the system uncertainties. As a continuous-time model to predict the system states at any time $t \in \mathbb{R}_+$, these two trajectories need to evolve continuously and simultaneously in the temporal domain. The two trajectories are realized with two major components, an ordinary differential equation (ODE) module parameterized with dynamic Diffusion Convolution, named as *dynDC-ODE*, and a stochastic differential equation (SDE) module. The former one encodes the stable parts and the topological information into the hidden features $H_t \in \mathbb{R}^{|\mathcal{V}| \times d_h} (t \in \mathbb{R}_+)$ while the latter one embeds the system uncertainties into the stochastic latent states $Z_t \in \mathbb{R}^{|V| \times d_z} (t \in \mathbb{R}_+)$. Although a single ODE

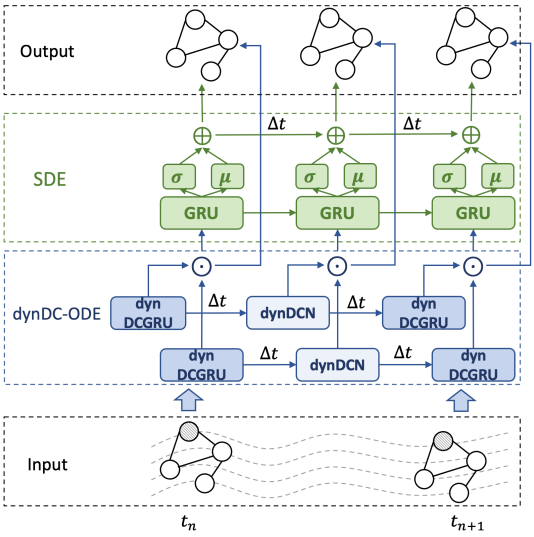

Fig. 1. The architecture of our proposed AGGDN.

or SDE is a continuous model, the observations are discrete, and the propagation within the interval between two observations always rely on the preceding observed data. As a result, the hidden state trajectory only makes an update at an observation point, and thus changes abruptly. To address this issue, in AGGDN, the ODE module first gives an approximation of H_t about the underlying hidden state, which is used to modulate the SDE module so that the latent state Z_t has a continuous dependency on the data, which in turn further refines the coarse hidden features H_t to get the final accurate output.

3.2 Component – Improved dynDC-ODE Module

Given an observation at a time instant t_n, our dynDC-ODE module extracts the topological relations of the graph and the stable properties of signals, which are embed into the continuous-time features H_{t_n}. The dynDC-ODE module has two functions: one is to employ a nonlinear encoder $G(\cdot)$ to directly integrate the information from data $\mathcal{O}_{t_n}$ observed at t_n, and the other is to apply a propagation function $F(\cdot)$ to update the features H_t within the interval (t_{n-1}, t_n).

Specifically, for the encoder $G(\cdot)$, we adopt the Diffusion Convolution Gated Recurrent Unit (DCGRU) [20] to integrate information from multi-hop neighbors for each node in the graph, which can be expressed as

$$H_{t_n} = G(H_{t_n-\delta t}, \mathcal{O}_{t_n}, A), \tag{2}$$

where A is the binary adjacency matrix of the graph. To capture the different impacts of neighbors during the integration, we learn another weight matrix W_A during the training and use the element-wise multiplication of two matrices $W_A \odot A$ to replace A in (2), forming our *dynamic Diffusion Convolution Unit (dynDCGRU)*. Therefore, at the observation time t_n, we can rewrite (2) as

$$H_{t_n} = G(H_{t_n-\delta t}, \mathcal{O}_{t_n}, W_A \odot A), \tag{3}$$

For the propagation function $F(\cdot)$, we parameterize the dynamics of hidden features during the interval (t_{n-1}, t_n) using ODEs:

$$\frac{dH_t}{dt} = F(H_t, W_A \odot A),$$
$$H_t = H_{t_{n-1}} + \int_{t_{n-1}}^{t} F(H_\tau, W_A \odot A)d\tau, \tag{4}$$

where the first-order derivative $F(\cdot)$ is formed through our *dynamic Diffusion Convolution Networks (dynDCN)*. For simplicity, we compute the integration in (4) by Euler Method, which can be expressed as

$$H_t = H_{t-\delta t} + F(H_{t-\delta t}, W_A \odot A)\delta t, \tag{5}$$

where δt is the propagation step size of the ODE module during the interval.

The observable data $\mathcal{O}_{t_n} = \mathcal{X}_{t_n} \odot \mathcal{M}_{t_n}$ are often irregular in the spatial domain due to the sample missing caused by sensor malfunction or transmission loss frequently appearing in the real-world networked system. To make our model adaptive to different missing locations (i.e., data loss from different nodes), we further introduce a *soft-masking function* into our dynDC-ODE module. The hidden feature H_t is then composed of a feature factor $H_{t,f}$ to extract the information and properties of the network, and a masking factor $H_{t,m} \in (0,1)^{|\mathcal{V}| \times d_h}$ to modulate the values in the feature. So the final feature would be

$$H_t = \rho(H_{t,m}W_m + b_m) \odot H_{t,f}, \tag{6}$$

where $\rho(\cdot)$ denotes the nonlinear activation function, W_m and b_m are the weight and bias parameters of a feed-forward network. Both $H_{t,f}$ and $H_{t,m}$ are learnt from the observed input and processed with (3) and (5). During the interval (t_{n-1}, t_n), the two factors will be updated by dynDC-ODEs as

$$H_{t,f} = H_{t-\delta t,f} + F_f(H_{t-\delta t,f}, W_A \odot A)\delta t,$$
$$H_{t,m} = H_{t-\delta t,m} + F_m(H_{t-\delta t,m}, W_A \odot A)\delta t, \tag{7}$$

At an observation time t_n, the new information from input data will be integrated by encoders as

$$H_{t_n,f} = G_f(H_{t_n-\delta t,f}, \mathcal{O}_{t_n}, W_A \odot A),$$
$$H_{t_n.m} = G_m(H_{t_n-\delta t,m}, \mathcal{O}_{t_n}, W_A \odot A). \tag{8}$$

3.3 Component – SDE Module

In many real-world networked systems, the observations are influenced by the process uncertainties. For example, in the case of a migrogrid network, the uncertainty in the control signal of distributed energy resource (DER) will cause oscillation in the current flows within the microgrid. To capture such uncertainties, we introduce a random latent state Z_t and parameterize it with a nonlinear SDE on the graph:

$$dZ_t = \mu(Z_t, H_{\leq t})dt + \sigma(H_{\leq t})dB_t,$$
$$Z_t = Z_{t_{n-1}} + \int_{t_{n-1}}^{t} \mu(Z_\tau, H_{\leq \tau})d\tau + \int_{t_{n-1}}^{t} \sigma(H_{\leq \tau})dB_\tau, \tag{9}$$

where μ and σ are the drift and diffusion functions respectively, B_t represents the standard Brownian motion. We define $\mu(\cdot)$ as the function of the current latent state Z_t and historical hidden features $H_{\leq t}$. However, $\sigma(\cdot)$ is only a function of the historical ODE features $H_{\leq t}$, as including Z_t also into $\sigma(\cdot)$ will bring additional noise term into the gradient computation [26] and make the training more difficult. Similar to the computation in dynDC-ODE, we compute the integration in (9) using Euler-Maruyama Method for simplicity:

$$Z_t = Z_{t-\delta t} + \mu(Z_{t-\delta t}, H_{\leq t})\delta t + \sqrt{\delta t}\sigma(H_{\leq t})\epsilon_t, \tag{10}$$

where δt is the step size and $\epsilon_t \in \mathcal{N}(0,1)$ is the standard Gaussian noise. In implementation, since the hidden features H_t extracted through dynDC-ODE have already captured the spatial topological relations within the network graph, we simply use a Gated Recurrent Unit (GRU) to integrate the historical information $H_{\leq t}$ of the ODE feature trajectory. Similarly, the drift function $\mu(\cdot)$ and the diffusion function $\sigma(\cdot)$ in (10) are also parameterized with Dense Neural Networks (DNNs) on the GRU features.

After computing hidden features H_t and latent states Z_t, the dynamic system states can be predicted with a trajectory of data dynamics and a residual term

$$\hat{\mathcal{X}}_t = N_{\hat{X}}(H_t) + N_{\hat{X}}^{(res)}(H_t, Z_t), \tag{11}$$

where $N_{\hat{X}}(\cdot)$ is the function of H_t to predict the smooth and stable trend of the signals while $N_{\hat{X}}^{(res)}(\cdot)$ incorporates the latent state to estimate the residual stochastic variations. Both $N_{\hat{X}}(\cdot)$ and $N_{\hat{X}}^{(res)}(\cdot)$ are implemented by simple DNNs.

3.4 Training Details

Since our model incorporates the latent states to model the system uncertainty, the log-likelihood in (1) of a single data instance $\mathcal{O}$, under an adjacency matrix A, can be rewritten as

$$\begin{aligned}\mathcal{L}_{ll}(\mathcal{O}) &= \sum_{n=1}^{N} \mathcal{M}_{t_n} \otimes \log P_{\mathcal{G}}(\mathcal{X}_{t_n} | \mathcal{O}_{t_0:t_{n-1}}, A) \\ &= \sum_{n=1}^{N} \mathcal{M}_{t_n} \otimes \log \int P_{\mathcal{G}}(\mathcal{X}_{t_n} | Z_{t_n}, \mathcal{O}_{t_0:t_{n-1}}, A) \\ &\qquad \times P_{\mathcal{G}}(Z_{t_n} | \mathcal{O}_{t_0:t_{n-1}}, A) dZ_{t_n},\end{aligned} \tag{12}$$

where $P_{\mathcal{G}}(Z_{t_n} | \mathcal{O}_{t_0:t_{n-1}}, A)$ is the conditional distribution of the latent states induced by SDE module while $P_{\mathcal{G}}(\mathcal{X}_{t_n} | Z_{t_n}, \mathcal{O}_{t_0:t_{n-1}}, A)$ is the conditional distribution of the observation.

In general, $P_{\mathcal{G}}(Z_{t_n} | \mathcal{O}_{t_0:t_{n-1}}, A)$ does not have a closed-form solution as Z_t is computed by the integration of a nonlinear SDE, thus $\mathcal{L}_{ll}(\cdot)$ does not have a closed-form solution either. However, after we synthesize a trajectory of the latent states $\{Z_t\}$ by (10), an alternative is to simplify the log-likelihood in (12) as the logarithm of the conditional distribution of observations:

$$\mathcal{L}_{con}(\mathcal{O}|\{Z_t\}) = \sum_{n=1}^{N} \mathcal{M}_{t_n} \otimes \log P_{\mathcal{G}}(\mathcal{X}_{t_n}|Z_{t_n}, \mathcal{O}_{t_0:t_{n-1}}, A), \tag{13}$$

where only one latent trajectory $\{Z_t\}$ is enough for estimating $P_{\mathcal{G}}(Z_{t_n}|\mathcal{O}_{t_0:t_{n-1}}, A)$ by Monte-Carlo Method. However, there is still a large difference between (13) and (12), which will compromise the training quality of our model. Conventionally, a state-space model with SDE is usually trained with the variational inference, where an evidence lower bound of the log-likelihood is given through an auxiliary inference model. The performance of variational inference is highly dependent on the accuracy of the inference model, and it often requires a Monte-Carlo Method with multiple samples to reduce the variance of the evidence lower bound. But in many real-world applications, if there is a large number of nodes and signals, an accurate inference model is hard to define and a Monte-Carlo Method running over a large number of samples is computationally expensive.

To efficiently train our model and avoid the drawbacks of variational inference method, we utilize the Wasserstein adversarial training objective:

$$\begin{aligned}\mathcal{L}_{adv}(\mathcal{G}, \mathcal{F}) = \mathbb{E}_{\{\mathcal{X}_{t_n}, \mathcal{M}_{t_n}\} \in D}[\mathcal{F}(\{\mathcal{X}_{t_n} \odot \mathcal{M}_{t_n}\}) \\ - \mathbb{E}_{\epsilon_t \in \mathcal{N}(0,1)} \mathcal{F}(\{\hat{\mathcal{X}}_{t_n} \odot \mathcal{M}_{t_n}\})],\end{aligned} \tag{14}$$

where $\mathcal{G}$ is our proposed model and $\mathcal{F}$ is the discriminator in the adversarial training. D is the dataset while $\{\hat{\mathcal{X}}_{t_n}\}$ is the set of predicted states given by our model, and $\{\epsilon_t\}$ is the uncertainty term in the SDE module. Based on the adversarial training, the discriminator is optimized by maximizing (14) while our model is optimized by minimizing the combination of the conditional log-likelihood in (13) and the adversarial loss in (14), i.e.

$$\mathcal{G}_* = arg \min_{\mathcal{G}} \Big(\lambda \mathcal{L}_{adv}(\mathcal{G}, \mathcal{F}) - \mathbb{E}_{\mathcal{O} \in D} \mathcal{L}_{con}(\mathcal{O}|\{Z_t\}) \Big), \tag{15}$$

$$\mathcal{F}_* = arg \max_{\mathcal{F}} \Big(\mathcal{L}_{adv}(\mathcal{G}, \mathcal{F}) \Big), \tag{16}$$

where λ is the weight coefficient.

Discriminator: The discriminator in our design is a function to convert the sporadic observation sequence, including the original one $\{\mathcal{O}_{t_n} = \mathcal{X}_{t_n} \odot \mathcal{M}_{t_n}\}$ and the predicted one $\{\hat{\mathcal{O}}_{t_n} = \hat{\mathcal{X}}_{t_n} \odot \mathcal{M}_{t_n}\}$, into a scalar in $\mathbb{R}$. Actually, our proposed model is independent of the choice of the discriminator, but for simplicity, we incorporate a DCGRU to extract features from inputs. The extracted feature in the last frame is applied to compute the scalar for each load in the microgrid. The output of the discriminator is given as the summation of all these scalar values. To meet the K-Lipschitz requirement in the Wasserstein adversarial objective, we apply Spectral Normalization (SN) to all the weight parameters in DCGRU and the output layer in the discriminator.

4 Experiments

We demonstrate the effectiveness of each component in our proposed model and show its robustness under different conditions through experiments on several datasets.

4.1 Experimental Setups

Datasets We first introduce the benchmarks and the pre-processing of data as follows:

IEEE33-Nodes: As a typical microgrid instance, the IEEE-33 Bus System contains 5 electricity sources and 28 load nodes. The 2-dimentional DQ current signals going through each node are generated using a hardware-based tool RTDS [33]. We collect 50 trajectories with signals sampled at the interval of 3 milliseconds for 21 different network topologies. For each trajectory, we split the sequence of samples into segments of 100 frames for training and testing. Before fed into the model, all data are normalized with the mean and the standard derivation in the temporal domain of the corresponding node.

METR-LA [20]: This traffic dataset records the speeds of vehicles on the highway of Los Angeles County. The data are collected by 207 sensors every 5 minutes. We split the data samples into segments of 36 frames and normalize the samples with the global means and standard derivations of all sensors in the temporal domain.

PEMS-BAY [20]: Similar to METR-LA, this is also a traffic dataset collected by 325 sensors every 5 minutes in the Bay Area. The pre-processing is the same as we did to METR-LA.

Evaluation Metrics We compare the values predicted with our proposed model based on sporadic observations and fully observable ground-true data using the metrics Mean Absolute Error (MAE, $\downarrow$), Root-Mean-Square Error (RMSE, $\downarrow$), and Mean Absolute Percentage Error (MAPE, $\downarrow$).

Implementation Details Our model consists two modules: one dynDC-ODE module and one SDE module. In the dynDC-ODE module, we use a dynDCGRU to encode the input signals into 32-dimensional hidden features and a 1-layer dynDCN with random-walk range $K = 3$ to parameterize the ODE. In the SDE module, a simple GRU cell and a 2-layer DNN are used to convert the hidden features from dynDC-ODE to the corresponding 2-dimensional latent states. The propagation step of both dynDC-ODE and SDE is $\delta t = 0.1\Delta T$, where ΔT is the sampling interval of input data. For each trajectory in the datasets, we synthesize 25 sporadic observation sequences with the random selection, which we will give details in the following part. After that, all data are split into training/validation/test sets with ratio 0.8/0.1/0.1. All models are trained by the ADAM optimizer with the learning rates $[10^{-2}, 10^{-3}, 10^{-4}]$, where 100 epochs are trained for each rate. λ in the adversarial training objective of (15) is 1.0.

4.2 Experiment Results

Overall Performance To synthesize the scenario of the sporadic observations, we randomly select a ratio p_t of the data frames in the temporal dimension as observed data. For each selected frame, we further assume only signals from

Datasets	IEEE33-Nodes			METR-LA			PEMS-BAY		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Discrete									
STGCN [47]	0.0812	0.1273	18.18%	0.2290	0.4375	37.58%	0.2499	0.4468	49.63%
Graph-GRU [36, 48]	0.0349	0.0643	9.65%	0.1978	0.4226	33.45%	0.1925	0.3725	41.17%
Continuous									
Graph-ODE-RNN [32]	0.0306	0.0578	8.36%	0.1918	0.4217	32.73%	0.1774	0.3605	37.34%
Graph-GRU-ODE [5]	0.0313	0.0607	8.49%	0.1947	0.4322	32.90%	0.1726	0.3488	37.44%
Ours									
AGGDN	**0.0243**	**0.0457**	**7.03%**	**0.1612**	**0.3480**	**30.41%**	**0.1489**	**0.2739**	**35.32%**

Table 1. Testing Performance of different models on various datasets ($p_t = 0.5, p_s = 0.8$)

p_s of the nodes are observed. Therefore, the data fed into our model has only $p_t \times p_s$ values remained, which can be regarded as sparse and also irregular in both temporal and spatial domains due to the random selection. Table 1 shows the performance comparison between our method and some other representative literature works on ($p_t = 0.5, p_s = 0.8$) case. From the table, we can see that our proposed method has a better performance on all datasets.

Ablation Study We perform experiments on the IEEE33-Nodes dataset to demonstrate the effectiveness of each component in our proposed model, including the *continuous-time modeling*, the *dynamic diffusion convolution* and the *soft-masking function* in the dynDC-ODE module, the *usage of the SDE module* and the *adversarial training strategy*.

Continuous-time Modeling: From Table 1, we can see that no matter our proposed model or other continuous-time models are superior to traditional discrete-time models, which only update features at the observation time instants. Continuous models make the updates $\Delta T/\delta t$ times between two neighboring observations, where the propagation step δt is much smaller than the sampling interval ΔT (we set $\delta t = 0.1\Delta T$ in the experiment). When used with graph convolutions, the integrated information from adjacent nodes will continuously help correct and update the states of the current node. Therefore, even though we use the simplest Euler-Method to approximately calculate the integration, the predicted trajectory fits the original one better than the trajectory provided by discrete models. Moreover, since the propagation step can be arbitrary small, the continuous-time model can provide predictions at any time to enable timely control, rather than only discretely on observation time points.

Dynamic Diffusion Convolution: Unlike most literature works using Graph Convolution (GC), Diffusion Convolution (DC) in our model integrates the information from neighbors within K hops (we set $K = 3$). This is especially helpful when data are sparse and superior to traditional graph convolution, which only collect and share information among 1-hop neighbors. Besides, our model learns the impacts of different neighbors and puts in weights in the aggregation, forming the dynamic Diffusion Convolution (dynDC), so that the information integration is more effective and accurate. To illustrate the effectiveness, we conduct the corresponding experiments and the results are shown in Table 2.

	GC [17]			DC [20]			dynDC		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Graph-GRU	0.0349	0.0643	9.65%	0.0341	0.0622	9.64%	**0.0319**	**0.0571**	**9.13**%
Graph-ODE-RNN	0.0306	0.0578	8.36%	0.0279	0.0522	7.83%	**0.0264**	**0.0504**	**7.49**%
Graph-GRU-ODE	0.0313	0.0607	8.49%	0.0298	0.0567	8.18%	**0.0272**	**0.0519**	**7.65**%

Table 2. Performance comparison among models using graph convolution (GC), diffusion convolution (DC) and dynamic diffusion convolution (dynDC) on IEEE33-Nodes ($p_t = 0.5, p_s = 0.8$).

Soft-masking Function: As described in Sec. 3, we introduce a soft-masking function to modulate the hidden features in our dynDC-ODE module for better adapting to the missing data cases. The comparison results in Table 3 have proved the role of such a design.

	MAE	RMSE	MAPE
AGGDN (w/o soft-masking)	0.0264	0.0504	7.49%
AGGDN (w/ soft-masking)	**0.0250**	**0.0471**	**7.21**%

Table 3. Performance comparison of our dynDC-ODE module with & without the soft-masking function on IEEE33-Nodes ($p_t = 0.5, p_s = 0.8$).

SDE Module: To better capture uncertainties existing in all the real-world systems, the stochastic modules, i.e. SDE, is included in our model. Different from ODE models which make strong Gaussian assumptions about the data distribution, the real distribution is learnt by Monte-Carlo sampling process within the SDE propagation. We also implement a simplified version, AGGDN(ODE), that does not have the SDE part for comparison. The experiments results based on both schemes running on the same data are shown in Table 4, and the improvement brought by the SDE module is obvious.

	MAE	RMSE	MAPE
AGGDN (ODE)	0.0250	0.0471	7.21%
AGGDN (full)	**0.0243**	**0.0457**	**7.03**%

Table 4. Performance comparison of our full model and the simplified model without SDE on IEEE33-Nodes ($p_t = 0.5, p_s = 0.8$).

Adversarial Training: Since we incorporate stochastic terms in our model, the training process becomes more difficult. To better train our model, we utilize an adversarial training strategy to avoid the possible gradient explosion problem. For reference, we also attach the comparison results before and after using the adversarial training in Table 5.

Robustness Study To test the generalization capability of our model, we also conduct extra experiments using the models trained by IEEE33-Node data with the observation ratio ($p_t = 0.4, p_s = 0.6$), and further evaluate the performance on data by varying the observation ratios. Besides our AGGDN, our dynamic Diffusion Convolution (dynDC) also helps improve the baselines and the results are plotted in Fig. 2. We can see that, compared with continuous-time models,

	MAE	RMSE	MAPE
AGGDN (w/o adversarial training)	0.0250	0.0473	7.20%
AGGDN (w/ adversarial training)	**0.0243**	**0.0457**	**7.03%**

Table 5. Performance comparison of our model on IEEE33-Nodes ($p_t = 0.5, p_s = 0.8$) before & after using adversarial training.

the Graph-GRU cannot adapt well to different observation ratios and its performance even becomes worse when more data are observed. Compared with other continuous-time models, our AGGDN consistently performs better, especially when the observation ratio is small. When $p_s = 1.0$, the simplified version AGGDN(ODE) is a little bit better than the full version AGGDN(full) since the effect of uncertainties brought by data missing is reduced and all the data properties and topological relation can be directly inferred from data.

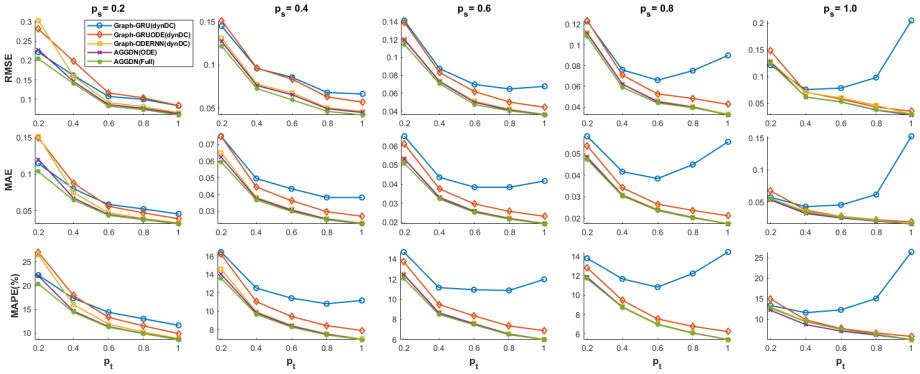

Fig. 2. Performance of various models with respect to different ratios of observations.

5 Conclusion

We propose a novel continuous-time stochastic model called AGGDN to model the dynamics of real-world networked systems from sporadic observations. Our model adopts a compounded ODE-SDE structure to capture the topological information and the signal properties on the graph while taking the underlying uncertainties in the system into consideration. The ODE component provides an approximate estimation of the signal, which further modulates SDE to refine the ODE result to provide a more accurate output. To better model the interactions among nodes, we propose a dynDC-ODE module with enhanced diffusion convolutions to learn the impact of different nodes during information integration. Besides, we introduce a soft-masking function to make our model adapt to the sparse and irregular data cases. To address the challenge of training with SDE module, the Wasserstein adversarial objective is incorporated. Experimental evaluations demonstrate that our model is effective in the state prediction and outperforms previous works on partial-observable networked systems such as microgrid and traffic networks.

Acknowledgments This work was supported in part by NSF under the award number ITE 2134840 and the U.S. DOE's Office of EERE under the Solar Energy Technologies Office Award Number 38456.

References

1. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems 31 (2018)
2. Choi, J., Choi, H., Hwang, J., Park, N.: Graph neural controlled differential equations for traffic forecasting (2021)
3. Chu, H., Li, D., Acuna, D., Kar, A., Shugrina, M., Wei, X., Liu, M.Y., Torralba, A., Fidler, S.: Neural turtle graphics for modeling city road layouts. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
4. Cui, Z., Lin, L., Pu, Z., Wang, Y.: Graph markov network for traffic forecasting with missing data. Transportation Research Part C: Emerging Technologies **117** (2020)
5. De Brouwer, E., Simm, J., Arany, A., Moreau, Y.: GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. In: Advances in Neural Information Processing Systems 32, pp. 7379–7390 (2019)
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems 29 (2016)
7. Diao, Z., Wang, X., Zhang, D., Liu, Y., Xie, K., He, S.: Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI-19) (Feb 2019)
8. Fang, Z., Long, Q., Song, G., Xie, K.: Spatial-temporal graph ODE networks for traffic flow forecasting. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery &amp Data Mining. ACM (aug 2021). https://doi.org/10.1145/3447548.3467430, https://doi.org/10.1145%2F3447548.3467430
9. Goyal, P., Chhetri, S.R., Canedo, A.: dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. arXiv **abs/1809.02657** (2018)
10. Goyal, P., Kamra, N., He, X., Liu, Y.: Dyngem: Deep embedding method for dynamic graphs. arXiv **abs/1805.11273** (2018)
11. Hajiramezanali, E., Hasanzadeh, A., Duffield, N., Narayanan, K.R., Zhou, M., Qian, X.: Variational graph recurrent neural networks. CoRR **abs/1908.09710** (2019), http://arxiv.org/abs/1908.09710
12. Huang, Z., Sun, Y., Wang, W.: Learning continuous system dynamics from irregularly-sampled partial observations. In: Advances in Neural Information Processing Systems 33 (2020)
13. Ioannidis, V.N., Marques, A.G., Giannakis, G.B.: A recurrent graph neural network for multi-relational data. In: 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8157–8161 (2019)
14. Jin, Y., JáJá, J.F.: Learning graph-level representations with gated recurrent neural networks. arXiv **abs/1805.07683** (2018)
15. Kidger, P., Morrill, J., Foster, J., Lyons, T.: Neural controlled differential equations for irregular time series (2020)
16. Kipf, T., Fetaya, E., Wang, K.C., Welling, M., Zemel, R.: Neural relational inference for interacting systems. In: Proceedings of the 35th International Conference on Machine Learning. pp. 2688–2697 (2018)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)

18. Kong, L., Sun, J., Zhang, C.: SDE-Net: Equipping deep neural network with uncertainty estimates. In: Proceedings of the 37th International Conference on Machine Learning (2020)
19. Li, X., Wong, T.K.L., Chen, R.T.Q., Duvenaud, D.: Scalable gradients for stochastic differential equations. In: 23rd International Conference on Artificial Intelligence and Statistics. pp. 3870–3882 (Aug 2020)
20. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations (ICLR) (2018)
21. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S.: Gated graph sequence neural networks. In: International Conference on Learning Representations (ICLR) (2016)
22. Li, Y., Vinyals, O., Dyer, C., Pascanu, R., Battaglia, P.W.: Learning deep generative models of graphs. arXiv **abs/1803.03324** (2018)
23. Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D.K., Urtasun, R., Zemel, R.: Efficient graph generation with graph recurrent attention networks. In: Advances in Neural Information Processing Systems 32. pp. 4255–4265 (2019)
24. Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., Hsieh, C.J.: Neural sde: Stabilizing neural ode networks with stochastic noise (2019). https://doi.org/10.48550/ARXIV.1906.02355, https://arxiv.org/abs/1906.02355
25. Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., Hsieh, C.J.: How does noise help robustness? explanation and exploration under the neural sde framework. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
26. Liu, Y., Xing, Y., Yang, X., Wang, X., Shi, J., Jin, D., Chen, Z.: Learning continuous-time dynamics by stochastic differential networks. ArXiv **abs/2006.06145** (2020)
27. Liu, Y., Xing, Y., Yang, X., Wang, X., Shi, J., Jin, D., Chen, Z., Wu, J.: Continuous-time stochastic differential networks for irregular time series modeling. In: Mantoro, T., Lee, M., Ayu, M.A., Wong, K.W., Hidayanto, A.N. (eds.) Neural Information Processing. pp. 343–351. Springer International Publishing, Cham (2021)
28. Manessi, F., Rozza, A., Manzo, M.: Dynamic graph convolutional networks. Pattern Recognition **97** (2020)
29. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T.B., Leiserson, C.E.: EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20) (2020)
30. Peluchetti, S., Favaro, S.: Infinitely deep neural networks as diffusion processes. In: 23rd International Conference on Artificial Intelligence and Statistics. vol. 108, pp. 1126–1136 (Aug 2020)
31. Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., Park, J.: Graph neural ordinary differential equations. arXiv **abs/1911.07532** (2019)
32. Poli, M., Massaroli, S., Rabideau, C.M., Park, J., Yamashita, A., Asama, H., Park, J.: Continuous-depth neural models for dynamic graph prediction (2021). https://doi.org/10.48550/ARXIV.2106.11581, https://arxiv.org/abs/2106.11581
33. RTDS-Technologies-Inc.: Power hardware-in-the-loop (phil). *https://www.rtds.com/applications/power-hardware-in-the-loop/* (2022)
34. Rubanova, Y., Chen, T.Q., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. In: Advances in Neural Information Processing Systems 32, pp. 5321–5331 (2019)
35. Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., Battaglia, P.W.: Hamiltonian graph networks with ODE integrators. arXiv **abs/1909.12790** (2019)

36. Seo, Y., Defferrard, M., Vandergheynst, P., Bresson, X.: Structured sequence modeling with graph convolutional recurrent networks. In: The 25th International Conference on Neural Information Processing. pp. 362–373 (2018)
37. Shrivastava, H., Chen, X., Chen, B., Lan, G., Aluru, S., Liu, H., Song, L.: Glad: Learning sparse graph recovery. In: International Conference on Learning Representations (ICLR) (2020)
38. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
39. Sun, C., Karlsson, P., Wu, J., Tenenbaum, J.B., Murphy, K.: Predicting the present and future states of multi-agent systems from partially-observed visual data. In: International Conference on Learning Representations (ICLR) (2019)
40. Taheri, A., Gimpel, K., Berger-Wolf, T.: Learning graph representations with recurrent neural network autoencoders. KDD'18 Deep Learning Day (2018)
41. Tzen, B., Raginsky, M.: Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. ArXiv **abs/1905.09883** (2019)
42. Tzen, B., Raginsky, M.: Theoretical guarantees for sampling and inference in generative models with latent diffusions. In: 32nd Annual Conference on Learning Theory. vol. 99, pp. 3084–3114 (Jun 2019)
43. Yan, T., Zhang, H., Li, Z., Xia, Y.: Stochastic graph recurrent neural network (2020). https://doi.org/10.48550/ARXIV.2009.00538, https://arxiv.org/abs/2009.00538
44. Ying, R., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems 31. p. 4805–4815 (2018)
45. You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J.: GraphRNN: Generating realistic graphs with deep auto-regressive models. In: Proceedings of the 35th International Conference on Machine Learning. pp. 5708–5717 (2018)
46. Yu, B., Li, M., Zhang, J., Zhu, Z.: 3D graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting. arXiv **abs/1903.00919** (2019)
47. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: Proceedings of he 27th International Joint Conference on Artificial Intelligence (IJCAI) (2018)
48. Yu, B., Yin, H., Zhu, Z.: ST-UNet: A spatio-temporal u-network for graph-structured time series modeling. arXiv **abs/1903.05631** (2019)