Intelligent and Converged Networks ISSN 2708-6240
2024, 5(3): 237−247 DOI: 10.23919/ICN.2024.0017

Evolved differential model for sporadic graph time-series prediction

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

Abstract: Sensing signals of many real-world network systems, such as traffic network or microgrid, could be sparse and irregular in both spatial and temporal domains due to reasons such as cost reduction, noise corruption, or device malfunction. It is a fundamental but challenging problem to model the continuous dynamics of a system from the sporadic observations on the network of nodes, which is generally represented as a graph. In this paper, we propose a deep learning model called Evolved Differential Model (EDM) to model the continuous-time stochastic process from partial observations on graph. Our model incorporates diffusion convolutional network to parameterize continuous-time system dynamics by graph Ordinary Differential Equation (ODE) and graph Stochastic Differential Equation (SDE). The graph ODE is applied to accurately capture the spatial-temporal relation and extract hidden features from the data. The graph SDE can efficiently capture the underlying uncertainty of the network systems. With the recurrent ODE-SDE scheme, EDM can serve as an accurate online predictive model that is effective for either monitoring or analyzing the real-world networked objects. Through extensive experiments on several datasets, we demonstrate that EDM outperforms existing methods in online prediction tasks.

Key words: graph sequence prediction; sporadic time series; continuous model; stochastic model; differential equation

1 Introduction

Although a practical system operates continuously, we are only able to collect system states at certain discrete points. Many systems are networked with a number of components, but only sporadic observations can be made due to cost, unreliable communications, and device malfunction or failure.

Structure and topology of networked systems can be represented as graphs. Some example network systems are home networks, social networks, vehicle networks, communication networks, and power grids. Figure 1 shows a microgrid network with sporadic observations. If we represent the microgrid network as a graph, nodes are grid components (e.g., photovoltaic module or residential load) and edges indicate the cables between them. Signals on each node are bidirectional current flows that represent the electricity consumption or supply. We can also establish the topological relation of the current flows in the cables by switching the roles of nodes and edges in the original graph[1]. Current signals collected in either case could be sporadic.

In order to timely and effectively control the systems for reliable and intelligent operations, it is important to accurately predict the system states. In this paper, we study the challenging problem of modeling the graph dynamics in the scenario where the signals on nodes evolve continuously, and the observations of these signals are sporadic which are irregular in temporal and/or spatial domains.

It is highly non-trivial to learn the continuous-time structured dynamics from sporadic observations on a graph. The challenges mainly come from three sources.

• Yucheng Xing, Yingru Liu, Xuewen Yang, and Xin Wang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA. E-mail: {yucheng.xing, yingru.liu, xuewen.yang, x.wang}@stonybrook.edu.
• Jacqueline Wu is with the New York University, New York, NY 10012, USA. E-mail: jw7824@nyu.edu.
* To whom correspondence should be addressed.
Manuscript received: 2023-12-17; revised: 2024-04-01; accepted: 2024-04-25

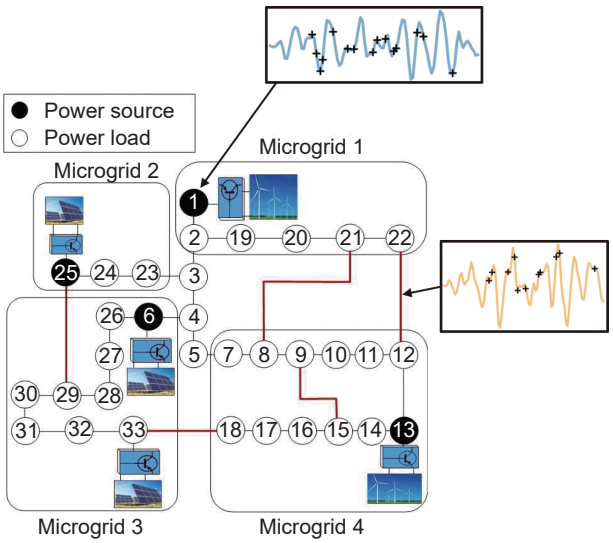

Fig. 1 Example of current flows in a microgrid network (the curves are the underlying dynamics of the signals and + denotes the discrete-time observations).

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 stochastic process. Last, as the output signals of network 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)[2–4], there are considerable number of studies on learning the dynamics of time series on graph[5–7], but most existing efforts merely consider discrete-time dynamics assuming systems are fully observed with complete data taken periodically. Although data missing is considered in Refs. [8, 9] for graph prediction, the scheme still considers discrete data samples and cannot 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[10–12], but ODE is only applied to learn the deterministic dynamics. With the process uncertainty of the system neglected, they are less capable of capturing the complicated distribution of observations in real-world systems.

We propose a continuous-time graph-based recurrent neural network, Evolved Differential Model (EDM), to capture the underlying dynamics on the graph structure from sporadic observations of node signals. EDM parameterizes the stochastic process of graph time series by Graph Ordinary Differential Equation (Graph-ODE) and Graph Stochastic Differential Equation (Graph-SDE). Inside the ODE component, 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 adapt to partial observations to better explore the spatial-temporal relation in the graph for extracting disentangled hidden features of partial observations. Furthermore, we incorporate the SDE component to efficiently capture the process uncertainty of the underlying system dynamics and conduct a more flexible parameterization of the data distribution.

The contributions of this paper include:

(1) We propose a neural ODE-SDE model based on diffusion graph convolution (GC) to accurately model the continuous-time process on a graph topology.

(2) We propose a soft-masking structure in the Graph-ODE that is adaptive to the partial observations of a network system to more effectively infer the missing data.

Extensive experiments on networked data demonstrate that EDM outperforms peer existing methods in sequential prediction with sporadic sample data.

The rest of this paper is organized as follows. The problem formulation and related works are described in Section 2. The detailed architectures of our model are proposed in Section 3. Extensive experiments are given in Section 4 and conclusions are made in Section 5.

2 Background

2.1 Problem formulation

2.1.1 Notation

We denote a graph with time-varying signals at nodes by $\boldsymbol{G}=\{\mathcal{V},\mathcal{E},\{\mathcal{X}_{t_n},\mathcal{M}_{t_n},t_n\}_{n=0}^{N}\}$, 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 multivariate 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.

2.1.2 Objective

Given a collection of data $D=\{\boldsymbol{G}^{(k)}\}_{k=1}^{|D|}$, where $\boldsymbol{G}^{(k)}$ is a data sequence introduced in Section 2.1.1 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:

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

where $\otimes$ is defined as the sum of element-wise product of two matrices, and $P_{\mathcal{G}}(\cdot)$ 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 Eq. (1) can be regarded as an unsupervised one.

2.2 Related works

2.2.1 Graph convolution

Graph convolution is the core operation of Graph Convolutional Network (GCN), an extension of the conventional Convolutional Neural Network (CNN) for structural objects. In the literature, various kinds of graph convolutional operators have been proposed[4, 13, 14]. To explore the relation among multi-hop neighbors in a graph, diffusion convolution was proposed in Ref. [14]:

$$(X\star A)_{W,b}=\sum_{k=0}^{K}(\mathcal{D}^{-1}A)^kXW_k+b_k \quad (2)$$

where A is the adjacent matrix of the graph, $\mathcal{D}$ is the diagonal degree matrix of A, and K is the number of transitions in the diffusion process. $W=\{W_k\}$ and $b=\{b_k\}$ are the trainable weight and bias parameters of the diffusion convolutional layer, respectively.

We construct our model based on diffusion convolution. Rather than only using a binary adjacency matrix A 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 straightforwardly adapted for other graph convolution methods.

2.2.2 Graph recurrent networks

To the best of our knowledge, most existing sequential graph models are proposed for discrete-time data[6, 15–19], 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[20–24] and feature learning[25–28]. DynGEM[29] and dyngraph2vec[30] 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[31, 32], especially under partial observations.

2.2.3 Neural differential equations

Neural Ordinary Differential Equation (NeuralODE) is first proposed in Ref. [33], where deep learning modules are incorporated 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 in Refs. [34, 35] by introducing a recurrent component that efficiently integrates the data information into the feature trajectories. There are some ODE-based studies[10–12, 36, 37] on learning 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 in the output, not to say that they can 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[38–43]. In Refs. [38–40], 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 Ref. [43] to compute the gradients for optimizing NeuralSDE. To the best of our knowledge, most NeuralSDE models[43–47] 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.

Compared to the literature work, we evolve a single neural differential equation to a compound model with the integration of NeuralODE and NeuralSDE into one infrastructure, which 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 Diffusion Convolutions (DCs). We further introduce an adaptive soft-masking scheme to work with the ODE components for a more accurate modeling in the presence of spatially irregular observations.

3 Evolved differential model

In this section, we propose a flexible continuous-time recurrent neural network, Evolved Differential Model (EDM), which is capable of learning the continuous graph dynamics from spatial-temporal irregular observation sequence. In the remaining of this section, we will introduce the model architecture of EDM and the structure of each model component in detail.

3.1 Network architecture

As shown in Fig. 2, EDM consists of a hidden feature trajectory and a latent state trajectory. The hidden feature is defined to integrate the topological relation in the graph and the latent state is introduced to capture the process uncertainty. To build a continuous time model that predicts the graph signals at any time $t \in \mathbb{R}_+$, the hidden feature and latent state trajectories should evolve continuously in the temporal domain. In order to efficiently extract the data information to embed into the hidden feature and latent state, EDM consists of two major components: a Graph-ODE module that encodes the disentangled factors and topological relation of the graph data into the hidden features $H_t \in \mathbb{R}^{|\mathcal{V}|\times d_h}$ $(t \in \mathbb{R}_+)$ and a Graph-SDE module that captures the system uncertainty and embeds it into the stochastic latent states $Z_t \in \mathbb{R}^{|\mathcal{V}|\times d_z}$ $(t \in \mathbb{R}_+)$.

The purpose of Graph-ODE module is to extract the topological relation and the signal property from the observations at discrete points, and embed them into the hidden feature H_t at any time. Therefore, we design the ODE module with two functions, the nonlinear mapping that directly integrates the information of data into the hidden feature at the observation time and the differential equation that updates the values of hidden feature over the interval between observations. Observations $\mathcal{O}_n = X_n \odot \mathcal{M}_n$ may be irregular in the spatial domain due to data loss, and there may be

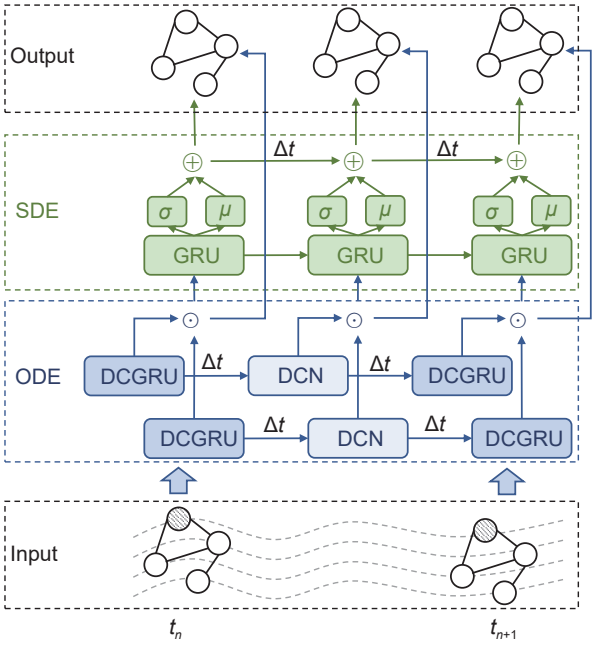

Fig. 2 Model architectures of EDM.

missing values from a node in a time frame. For EDM to adapt to the positions of missing data samples, we introduce a soft-masking function into the ODE module. The hidden feature H_t is composed of two factors, the feature factor $H_{t,f}$ that extracts the topological relation of the network and data property, and the masking factor $H_{t,m} \in (0,1)^{\mathcal{V}\times d_h}$ that modulates the values of the feature. It can be expressed as

$$H_t = \text{Sigmoid}(H_{t,m}W_m + b_m) \odot H_{t,f} \tag{3}$$

where Sigmoid$(\cdot)$ denotes the sigmoid activation function and $\odot$ denotes the element-wise multiplication. $\{W_m, b_m\}$ are the weight and bias parameters of the auxiliary feed-forward network, respectively. We parameterize the dynamics of $H_{t,f}$ and $H_{t,m}$ during the interval using ODEs:

$$\frac{\mathrm{d}H_{t,m}}{\mathrm{d}t} = F_m(H_{t,m}, A), \quad \frac{\mathrm{d}H_{t,f}}{dt} = F_f(H_{t,f}, A) \tag{4}$$

$$H_{t,m} = H_{t_{n-1},m} + \int_{t_{n-1}}^{t} F_m(H_{\tau,m}, A)\mathrm{d}\tau \tag{5}$$

$$H_{t,f} = H_{t_{n-1},f} + \int_{t_{n-1}}^{t} F_f(H_{\tau,f}, A)\mathrm{d}\tau \tag{6}$$

where $F_m(H_{t,m}, A)$ and $F_f(H_{t,f}, A)$ are the first-order derivatives computed by graph neural networks. In implementation, we compute the integration by Euler method:

$$H_{t,m} = H_{t-\Delta t,m} + F_m(H_{t-\Delta t,m}, A)\Delta t \tag{7}$$

$$H_{t,f} = H_{t-\Delta t,f} + F_f(H_{t-\Delta t,f}, A)\Delta t \tag{8}$$

where Δt is the step size. At an observation time t_n, we embed the information of the partial observations into the hidden feature by nonlinear mappings:

$$H_{t_n,m} = G_m(H_{t_n-\Delta t,m}, X_n \odot \mathcal{M}_n, A) \tag{9}$$

$$H_{t_n,f} = G_f(H_{t_n-\Delta t,f}, X_n \odot \mathcal{M}_n, A) \tag{10}$$

The soft-mask is introduced into the ODE module to modulate the extracted features with respect to partial observations in the graph to increase the prediction accuracy of system states.

Observations of many real-world networks are influenced by the process uncertainty inside the systems. For instance, the uncertainty in the Distributed Energy Resource (DER) control signal of a microgrid will cause large oscillation in current flows. We utilize a latent state Z_t, a random matrix, to embed the process uncertainty of the underlying dynamics of each node. In order to capture the complicated stochastic process of the network systems, we parameterize the latent state by a nonlinear SDE:

$$\mathrm{d}Z_t = \mu(Z_t, H_{\leqslant t})\mathrm{d}t + \sigma(H_{\leqslant t})\mathrm{d}B_t \tag{11}$$

$$Z_t = Z_{t_0} + \int_{t_0}^{t} \mu(Z_\tau, H_{\leqslant \tau})\mathrm{d}\tau + \int_{t_0}^{t} \sigma(H_{\leqslant \tau})\mathrm{d}B_\tau \tag{12}$$

where μ and σ are the drift and diffusion functions, respectively, and B_t denotes the standard Brownian motion. We define μ as the function of current latent state Z_t and historical ODE features $H_{\leqslant t}$. σ only takes historical ODE features as the input, as including latent state into σ will bring additional noise term into the gradient computation[45] and may be harmful to the training process.

We compute the value of Z_t using the Euler–Maruyama method:

$$Z_t = Z_{t-\Delta t} + \mu(Z_{t-\Delta t}, H_{\leqslant t})\Delta t + \sqrt{\Delta t}\sigma(H_{\leqslant t})\varepsilon_t \tag{13}$$

where Δt is the step size, $\varepsilon_t \sim \mathcal{N}(0,1)$ is the standard Gaussian noise.

After computing the hidden feature H_t and the latent state Z_t, the predictive states are the integration of a trajectory to capture the data evolution and a residual term:

$$\widehat{X}_t = \widehat{X}_t^0(H_t) + \widehat{X}_t^{(\text{res})}(H_t, Z_t) \tag{14}$$

where $\widehat{X}_t^0$ is the function of H_t to predict the smooth trend of the signal and $\widehat{X}_t^{(\text{res})}$ further incorporates the latent state to estimate the residual dynamic variations.

3.2 Component description

The architecture of EDM can be constructed by graph neural networks to effectively learn the complicated topological relation of nodes, extract disentangled factors from observations and track the temporal evolution of the dynamics from the irregular observation sequence. In our implementation, we build our model on diffusion convolution to capture the topological correlation of node states over multiple hops, however our model does not depend on the choice of graph convolutional operators and can be straightly adapted to use other graph convolution methods. The detailed designs of each component of

EDM are given as follows.

3.2.1 Graph-ODE modules

The Graph-ODE component consists of two parts: the derivative functions $\{F_m, F_f\}$ in Eqs. (7) and (8) and the mapping functions $\{G_m, G_f\}$ in Eqs. (9) and (10). Our mapping module is realized through a Diffusion Convolution Gated Recurrent Unit (DCGRU)[14]:

$$R = \text{Sigmoid}(([H_{t_n,*}, X_n \odot \mathcal{M}_n] \star A)_{W_R, b_R}) \tag{15}$$

$$U = \text{Sigmoid}(([H_{t_n,*}, X_n \odot \mathcal{M}_n] \star A)_{W_U, b_U}) \tag{16}$$

$$C = \tanh(([R \odot H_{t_n,*}, X_n \odot \mathcal{M}_n] \star A)_{W_C, b_C}) \tag{17}$$

$$H_{t_n,*} = U \odot H_{t_n,*} + (1 - U) \odot C \tag{18}$$

where $\star$ is the diffusion operator given in Eq. (2). Our derivative function is realized through a multi-layer Diffusion Convolutional Network (DCN) with diffusion convolution and ReLU activation function:

$$h_1 = \text{ReLU}((H_{t-\Delta t,*} \star A)_{W_{h_1}, b_{h_1}}) \tag{19}$$

$$h_l = \text{ReLU}((h_{l-1} \star A)_{W_{h_1}, b_{h_1}}), \quad 1 < l < L \tag{20}$$

$$F_* = (h_L \star A)_{W_{h_L}, b_{h_L}} \tag{21}$$

where L is the number of diffusion convolution layers. With the multi-hop property of diffusion convolution, the Graph-ODE component of our model is capable of exploring both the topological relation and the continuous-time dynamics in irregular graph time series.

3.2.2 SDE modules

The SDE component synthesizes a latent state trajectory from the ODE features to capture the process uncertainty. The ODE features are already computed by the Euler method with a fixed step-size Δt, and have embedded the spatial relation of the graph structure. Therefore, we simply introduce a conventional Gated Recurrent Unit (GRU) to integrate the historical part of the ODE feature trajectory. The drift μ and diffusion σ functions in Eq. (13) are defined as feed-forward networks on the GRU features.

3.2.3 Output modules

With the ODE features $\{H_t\}$ and SDE latent states $\{Z_t\}$, the output layers of EDM are simply two linear layers that compute $\widehat{X}_t^0(H_t)$ and $\widehat{X}_t^{(\text{res})}(H_t, Z_t)$ in Eq. (14).

4 Experiments

We conduct extensive experiments to evaluate the performance of several popular sporadic time series on graph.

4.1 Experiment setting

4.1.1 Datasets

We first describe the benchmarks and the corresponding data pre-processing as follows:

Microgrid-nodes: We generate microgrid data for IEEE 33-bus system using RTDS[48], a hardware-based tool to create the real-time power system data. RTDS is widely used by power industries and the data produced serve as the world's benchmark. The dynamics of microgrid networks are studied with 5 electricity source nodes and 28 load nodes under 21 different network structures. The signals from each microgrid node form a 2-dimensional vector which records the information of bidirectional current flows in the DQ domain going through the corresponding microgrid component. We take 50 trajectories from each network structure, with data observations made at the interval of $\Delta t = 3$ ms. For these trajectories, we divide each of them into segments with 100 frames for training and testing. The data collected from each node is further normalized by the global mean and standard deviation of the data in the temporal domain of that node.

Microgrid-edges: Bidirectional current flows in the cables between nodes form the multivariate signals at the edges of microgrid networks. To predict the signals on edges, we construct line graphs of microgrid networks by switching the roles of nodes and edges. Specifically, we consider two edges are connected if they have the same node as one of their ends. The pre-processing of the current flow signal is similar to that of Microgrid-Nodes.

METR-LA[14]: The traffic dataset METR-LA records the speeds of vehicles on the highway of Los Angeles County, CA, USA. The data are collected by 207 sensors every 5 minutes. We split the data samples into segments, each with 36 frames and also normalize the samples with the global mean and standard derivation of corresponding sensors in the temporal domain.

To synthesize the scenario of 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 also assume only p_s of the signal from a specific node is observed. Overall, we have only $p_t \times p_s$ data samples, which are sparse and also irregular in both spatial and temporal dimensions due to the random selection. In our experiments, we train deep learning models in two different configurations of (p_s, p_t): (i) $(p_s = 0.8, p_t = 0.5)$; (ii) $(p_s = 0.6, p_t = 0.4)$.

4.1.2 Baselines

We compare the performance of our models with both discrete-time and continuous-time methods. The discrete-time baselines are configured as follows:

STGCN[17]: This discrete-time model consists of several concatenated Spatio-Temporal Convolutional (ST-Conv) blocks, where a graph convolutional layer is introduced to explore the graph relation and two temporal convolutional layers are defined to explore the temporal dependency. A residual connection is also included to better train the convolutional network. In our experiments, STGCN has 2 ST-Conv blocks.

GCGRU[6, 15]: It is a variant of conventional GRU, where the linear operation in the conventional GRU is replaced by graph convolution. We use the graph convolutional operation in Ref. [4], which explores the 1-hop relation in a graph. We construct a baseline with single GCGRU layer and a linear output layer.

DCGRU[14]: Similar to GCGRU, it is a variant of GRU but with the linear operation replaced by the diffusion convolution in Eq. (2). The model construction of DCGRU is identical to that of GCGRU.

The continuous-time baselines include:

GCDE-GRU[11]: It is a GNN variant of ODE-RNN[34], with a GCN to learn the derivative of ODE and a GRU to integrate the information of observations.

GRU-ODE$^+$: GRU-ODE[35] is an ODE-based model whose structure is similar to ODE-RNN except that the derivative of the ODE is given by a differential variant of GRU. The original GRU-ODE is proposed for vector data, while we construct a GNN variant by replacing its linear operation with graph convolution. The configuration of GRU-ODE is identical to GCDE-GRU. To differentiate it from the original GRU-ODE, we call it GRU-ODE$^+$.

We consider two types of graph operations in GCDE-GRU and GRU-ODE$^+$: one is the first-order GC in Ref. [4], and the other is the Diffusion Convolution (DC) in Ref. [14]. We set the number of hops in diffusion convolution to 3.

In our model, the configurations of ODE components are identical to those in GCDE-GRU (DC). To demonstrate the effect of SDE component, we also implement a simplified model EDM (ODE) that does not have the SDE part.

4.1.3 Training setting

For each trajectory in the datasets, we synthesize 25 sporadic observation sequences. 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 Adaptive Moment Estimation (ADAM) optimizer[49] with the learning rate 0.001 and further fine-tuned with the learning rate 0.0001.

4.1.4 Metrics for evaluation

We evaluate the following metrics for the comparison between our model and the baselines: (i) Mean Absolute Error (MAE, ↓); (ii) Rooted Mean Square Error (RMSE, ↓); (iii) Mean Arctangent Absolute Percentage Error (MAAPE, ↓). All these metrics are evaluated between the predicted trajectories and the ground-true ones.

4.2 Quantitative evaluation

The performance of prediction in the testing sets is shown in Tables 1 and 2. STGCN fails to capture the dynamics from the partially observed data and has the worst performance in all cases. For a specific graph recurrent model, DC outperforms the first-order GC in most of the cases, especially when the missing rate is high, because DC effectively captures the multi-hop relation in the graph while GC only considers the correlation from the closest neighbors of the nodes. Our model is built upon GCDE-GRU (DC). To better demonstrate the effectiveness of our proposed components: soft-masking function and ODE-SDE structure, we prepare the corresponding two versions of our model: EDM (ODE) adds soft-masking based on

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

	Model	Microgrid-Nodes			Microgrid-Edges			METR-LA		
		MAE	RMSE	MAAPE	MAE	RMSE	MAAPE	MAE	RMSE	MAAPE
Discrete	STGCN	0.1097	0.1943	0.2166	0.0907	0.1771	0.1771	0.2870	0.5012	0.4069
	GCGRU	0.0650	0.1499	0.1456	0.0637	0.1499	0.1350	0.2319	0.4652	0.3342
	DCGRU	0.0667	0.1511	0.1522	0.0627	0.1487	0.1384	0.2367	0.4673	0.3415
Continuous	GCDE-GRU (GC)	0.0480	0.1368	0.1057	0.0452	0.1357	0.0945	0.2279	0.4674	0.3282
	GRU-ODE (GC)	0.0476	0.1388	0.1054	0.0448	0.1383	0.0939	0.2278	0.4653	0.3286
	GCDE-GRU (DC)	0.0482	0.1351	0.1073	0.0451	0.1347	0.0953	0.2207	0.4541	0.3248
	GRU-ODE (DC)	0.0474	0.1378	0.1050	0.0451	0.1391	0.0944	0.2240	0.4605	0.3270
Ours	EDM (ODE)	0.0451	0.1300	0.1025	0.0434	0.1311	0.0929	0.2190	0.4529	0.3222
	EDM (full)	**0.0437**	**0.1276**	**0.0999**	**0.0426**	**0.1300**	**0.0914**	**0.2168**	**0.4512**	**0.3204**

Table 2 Testing performance of different models on various datasets (p_s = 0.6, p_t=0.4).

	Model	Microgrid-Nodes			Microgrid-Edges			METR-LA		
		MAE	RMSE	MAAPE	MAE	RMSE	MAAPE	MAE	RMSE	MAAPE
Discrete	STGCN	0.1427	0.2565	0.2491	0.1167	0.2302	0.2165	0.3535	0.5733	0.4657
	GCGRU	0.0973	0.2197	0.1816	0.0893	0.2079	0.1732	0.2945	0.5318	0.4029
	DCGRU	0.0975	0.2176	0.1866	0.0901	0.2067	0.1795	0.2876	0.5248	0.3939
Continuous	GCDE-GRU (GC)	0.0768	0.2043	0.1393	0.0707	0.1942	0.1326	0.2839	0.5320	0.3838
	GRU-ODE (GC)	0.0781	0.2120	0.1403	0.0699	0.1974	0.1313	0.2767	0.5215	0.3794
	GCDE-GRU (DC)	0.0711	0.1944	0.1323	0.0676	0.1880	0.1298	0.2713	0.5114	0.3758
	GRU-ODE (DC)	0.0745	0.2051	0.1361	0.0685	0.1955	0.1294	0.2711	0.5145	0.3740
Ours	EDM (ODE)	0.0707	0.1943	0.1317	0.0661	0.1866	0.1270	0.2694	0.5101	0.3731
	EDM (full)	**0.0688**	**0.1892**	**0.1302**	**0.0653**	**0.1862**	**0.1255**	**0.2676**	**0.5093**	**0.3717**

GCDE-GRU (DC) while EDM (full) includes SDE module in further. With the adaptivity of the soft-masking structure, EDM (ODE) outperforms all the baselines, since it allows our model to learn how to integrate information from partial observation during training and adapt the aggregation weights according to the data missing states at inference phase. Furthermore, EDM (full) with both ODE and SDE components achieves the best prediction accuracy in all the dataset, as the SDE component efficiently embeds the process uncertainty.

5 Conclusion

We propose a continuous-time stochastic graph model called EDM to model the dynamics of real-world network systems from sporadic observations. Through extensive experiments, we demonstrate that the hybrid ODE-SDE scheme of EDM can efficiently and accurately model the stochastic process of the underlying mechanism from the sparse and irregular data of a networked system such as microgrid and traffic network. In our future works, we are going to extend our model for large-scale networks with hierarchical structures.

References

[1] X. Jiang, P. Ji, and S. Li, CensNet: Convolution with edge-node switching in graph neural networks, in *Proc. 28th Int. Joint Conf. Artificial Intelligence*, Macao, China, 2019, pp. 2656–2662.

[2] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in *Proc. 32nd Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4805–4815.

[3] M. Simonovsky and N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Honolulu, HI, USA, 2017, pp. 29–38.

[4] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *Proc. 2017 IEEE Int. Conf. Computer Vision* (*ICCV*), Venice, Italy, 2017.

[5] T. Kipf, E. Fetaya, K. C. Wang, M. Welling, and R. Zemel, Neural relational inference for interacting systems, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 2688–2697.

[6] B. Yu, H. Yin, and Z. Zhu, ST-UNet: A spatio-temporal U-network for graph-structured time series modeling, arXiv preprint arXiv: 1903.05631, 2019.

[7] B. Yu, M. Li, J. Zhang, and Z. Zhu, 3D graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting, arXiv preprint arXiv: 1903.00919, 2019.

[8] C. Sun, P. Karlsson, J. Wu, J. B. Tenenbaum, and K. Murphy, Predicting the present and future states of multi-agent systems from partially-observed visual data, in *Proc. 7th Int. Conf. Learning Representations* (*ICLR*), New Orleans, LA, USA, 2019.

[9] Z. Cui, L. Lin, Z. Pu, and Y. Wang, Graph Markov network for traffic forecasting with missing data, *Transp. Res. Part C Emerg. Technol.*, vol. 117, p. 102671, 2020.

[10] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. W. Battaglia, Hamiltonian graph networks with ODE integrators, arXiv preprint arXiv: 1909.12790, 2019.

[11] M. Poli, S. Massaroli, J. Park, A. Yamashita, H. Asama, J. Park, M. Poli, S. Massaroli, C. M. Rabideau, J. Park et al., Graph neural ordinary differential equations, arXiv preprint arXiv: 1911.07532, 2019.

[12] Z. Huang, Y. Sun, and W. Wang, Learning continuous system dynamics from irregularly-sampled partial observations, in *Proc. 34th Conf. Neural Information Processing Systems* (*NeurIPS 2020*), Vancouver, Canada, 2020, pp. 16177–16187.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, arXiv preprint arXiv: 1606.09375, 2016.

[14] Y. Li, R. Yu, C. Shahabi, and Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv preprint arXiv: 1707.01926, 2017.

[15] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, Structured sequence modeling with graph convolutional recurrent networks, in *Proc. 25th Int. Conf. Neural Information Processing* (*ICONIP 2018*), Siem Reap, Cambodia, 2018, pp. 362–373.

[16] F. Manessi, A. Rozza, and M. Manzo, Dynamic graph convolutional networks, *Pattern Recognit.*, vol. 97, p. 107000, 2020.

[17] B. Yu, H. Yin, and Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in *Proc. 27th Int. Joint Conf. Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 3634–3640.

[18] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting, in Proc. 32nd AAAI Conf. Artificial Intelligence, 31st Innovative Applications of Artificial Intelligence Conf., 9th AAAI Symp. Educational Advances in Artificial Intelligence, Honolulu, HI, USA, 2019, pp. 890–897.

[19] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, EvolveGCN: evolving graph convolutional networks for dynamic graphs, in *Proc. 34th AAAI Conf. Artificial Intelligence* (*AAAI-20*), New York, NY, USA, pp. 5363–5370.

[20] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, GraphRNN: generating realistic graphs with deep auto-regressive models, arXiv preprint arXiv: 1802.08773, 2018.

[21] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, Learning deep generative models of graphs, arXiv preprint arXiv: 1803.03324, 2018.

[22] R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D. K. Duvenaud, R. Urtasun, and R. Zemel, Efficient graph generation with graph recurrent attention networks, in *Proc. 33rd Conf. Neural Information Processing Systems* (*NeurIPS 2019*), Vancouver, Canada, 2019, pp. 4255–4265.

[23] H. Shrivastava, X. Chen, B. Chen, G. Lan, S. Aluru, H. Liu, and L. Song, Glad: Learning sparse graph recovery, in *Proc. 8th Int. Conf. Learning Representations* (*ICLR*), virtual, 2020.

[24] H. Chu, D. Li, D. Acuna, A. Kar, M. Shugrina, X. Wei, M. Y. Liu, A. Torralba, and S. Fidler, Neural turtle graphics for modeling city road layouts, in *Proc. IEEE/CVF Int. Conf. Computer Vision* (*ICCV*), Seoul, Republic of Korea, 2019, pp. 4521–4529.

[25] Y. Jin and J. F. J′aJ′a, Learning graph-level representations with gated recurrent neural networks, arXiv preprint arXiv: 1805.07683, 2018.

[26] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, Gated graph sequence neural networks, in *Proc. 4th Int. Conf. Learning Representations* (*ICLR*), San Juan, Puerto Rico, 2016.

[27] A. Taheri, K. Gimpel, and T. Berger-Wolf, Learning graph representations with recurrent neural network autoencoders, https://www.kdd.org/kdd2018/files/deep-

learning-day/DLDay18_paper_27.pdf, 2018.

[28] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, A recurrent graph neural network for multi-relational data, in *Proc. ICASSP 2019-2019 IEEE Int. Conf. Acoustics, Speech and Signal Processing* (*ICASSP*), Brighton, UK, 2019, pp. 8157–8161.

[29] P. Goyal, N. Kamra, X. He, and Y. Liu, DynGEM: deep embedding method for dynamic graphs, arXiv preprint arXiv: 1805.11273, 2018.

[30] P. Goyal, S. R. Chhetri, and A. Canedo, dyngraph2vec: Capturing network dynamics using dynamic graph representation learning, *Knowl. Based Syst.*, vol. 187, p. 104816, 2020.

[31] E. Hajiramezanali, A. Hasanzadeh, N. Duffield, K. R. Narayanan, M. Zhou, and X. Qian, Variational graph recurrent neural networks, arXiv preprint arXiv: 1908.09710, 2019.

[32] T. Yan, H. Zhang, Z. Li, and Y. Xia, Stochastic graph recurrent neural network, arXiv preprint arXiv: 2009.00538, 2020.

[33] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, Neural ordinary differential equations, arXiv preprint arXiv: 1806.07366, 2018.

[34] Y. Rubanova, T. Q. Chen, and D. K. Duvenaud, Latent ordinary differential equations for irregularly-sampled time series, in *Proc. 33rd Conf. Neural Information Processing Systems* (*NeurIPS 2019*), Vancouver, Canada, 2019, pp. 5321–5331.

[35] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, GRUODE- Bayes: Continuous modeling of sporadically-observed time series, in *Proc. 33rd Conf. Neural Information Processing Systems* (*NeurIPS 2019*), Vancouver, Canada, 2019, pp. 7379–7390.

[36] Z. Fang, Q. Long, G. Song, and K. Xie, Spatial-temporal graph ODE networks for traffic flow forecasting, in *Proc. 27th ACM SIGKDD Conf. Knowledge Discovery & Data Mining*, virtual, 2021.

[37] J. Choi, H. Choi, J. Hwang, and N. Park, Graph neural controlled differential equations for traffic forecasting, in *Proc. 36th AAAI Conference on Artificial Intelligence* (*AAAI-22*), virtual, 2022, pp. 6367–6374.

[38] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C. J. Hsieh, How does noise help robustness? explanation and exploration under the neural SDE framework, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Seattle, WA, USA, 2020, pp. 279–287.

[39] S. Peluchetti and S. Favaro, Infinitely deep neural networks as diffusion processes, in *Proc. 23rd Int. Conf. on Artificial Intelligence and Statistics*, virtual, 2020, pp. 1126–1136.

[40] L. Kong, J. Sun, and C. Zhang, SDE-Net: Equipping deep neural network with uncertainty estimates, in *Proc. 37th Int. Conf. Machine Learning*, virtual, 2020.

[41] B. Tzen and M. Raginsky, Theoretical guarantees for sampling and inference in generative models with latent diffusions, in *Proc. 32nd Annual Conf. Learning Theory*, Phoenix, AZ, USA, 2019, pp. 3084–3114.

[42] B. Tzen and M. Raginsky, Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit, arXiv preprint arXiv: 1905.09883, 2019.

[43] X. Li, T. K. L. Wong, R. T. Q. Chen, and D. Duvenaud, Scalable gradients for stochastic differential equations, in *Proc. 23rd Int. Conf. Artificial Intelligence and Statistics*, virtual, 2020, pp. 3870–3882.

[44] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C. J. Hsieh, Neural SDE: Stabilizing neural ODE networks with stochastic noise, arXiv preprint arXiv: 1906.02355, 2019.

[45] Y. Liu, Y. Xing, X. Yang, X. Wang, J. Shi, D. Jin, and Z. Chen, Learning continuous-time dynamics by stochastic differential networks, arXiv preprint arXiv: 2006.06145, 2020.

[46] Y. Liu, Y. Xing, X. Yang, X. Wang, J. Shi, D. Jin, Z. Chen, and J. Wu, Continuous-time stochastic differential networks for Irregular time series modeling, in *Proc. 28th Int. Conf. Neural Information Processing*, Bali, Indonesia, 2021, pp. 343–351,

[47] Y. Liu, Deep generative models for stochastic modeling of multivariate sequential data, Ph. D. dissertation, State University of New York at Stony Brook, USA, 2021.

[48] RTDS-Technologies-Inc., Power hardware-in-the loop (phil), https://www.rtds.com/applications/power-hardware-in-the-loop/, 2022.

[49] D. P. Kingma, J. Ba, and M. M. Hammad, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2014.

Yucheng Xing received the BS degrees in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2017. He is currently pursuing the PhD degree in electrical engineering with Stony Brook University, Stony Brook, NY, USA. His current research interests include the graph modeling and adaptive machine learning.

Jacqueline Wu is currently a student in New York University. Her research interests include big data analysis and machine learning.

Yingru Liu received the PhD degree in computer engineering from the Electrical and Computer Engineering Department, Stony Brook University, Stony Brook, NY, USA, in 2021. Before that, he received the BE degree in automation from the Department of Automation Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2015. His research interests include machine learning, time series, probabilistic graphical models, and adaptive control.

Xin Wang received the BS and MS degrees in telecommunications engineering and wireless communications engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1990 and 1993, respectively, and the PhD degree in electrical and computer engineering from Columbia University, New York, NY, USA, in 2001. She is currently an associate professor with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY, USA. Before joining Stony Brook, she was a member of technical staff in the area of mobile and wireless networking with Bell Labs Research, Lucent Technologies, Murray Hill, NJ, USA, and an assistant professor with the Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY, USA. Her research interests include big data analysis and machine learning, wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection. Dr. Wang achieved the U.S. National Science Foundation Career Award in 2005 and the ONR Challenge Award in 2010. She has served in executive committee and technical committee of numerous conferences and funding review panels, and served as the associate editor of *IEEE Transactions on Mobile Computing*.

Xuewen Yang, an accomplished scholar specializing in multimodel learning and natural language processing, is currently a senior researcher at InnoPeak Technology, Palo Alto, USA. He received the BS and MS degrees from Xi'an Jiaotong University, China, and another MS degree from Ecole Centrale Marseille, France. He received the PhD degree from Stony Brook University, New York, USA in 2021. With a focus on bridging theory and practice, Dr. Yang is driving technological innovation in computational systems through his research.