Evolutionary Graph Normalizing Flows

Daheng Wang, Tong Zhao, Nitesh V. Chawla, Meng Jiang
Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
{dwang8,tzhao2,nchawla,mjiang2}@nd.edu

ABSTRACT

Graph representation learning aims at preserving structural and attributed information in latent representations. It has been studied mostly in the setting of static graph. In this work, we propose a novel approach for representation learning over evolutionary attributed graph using the tool of normalizing flows for exact density estimation. Our approach has three components: (1) a time-aware graph neural component for aggregating graph information at each time step, (2) an adapted graph recurrent component for updating graph temporal contexts, and (3) a conditional normalizing flows component for capturing the evolution of node representations in latent space along time. The third component has two sub-models of normalizing flows. One is used to capture the distribution of node representations of arbitrary complexity by considering graph temporal contexts as conditions. It learns invertible transformations to map node representations into simple priors conditioning on temporal contexts. The other one is dedicated to capture the evolutionary patterns of prior distributions. Extensive experiments show the proposed approach can outperform competitive baselines by a significant margin for link prediction on future graph structure.

1 INTRODUCTION

Graph representation learning has been widely studied for learning node latent representations of a static graph. However, real graphs are dynamic because the graph structure and attributes information are both constantly changing. Evolutionary attributed graph often exhibit complex temporal patterns of the graph's evolution over time. For example, social network users establish and/or remove links between each other via daily behaviors such as following, mentioning and replying. And, the user's attributes such as textual features from her generated content are also changing. The updates of graph structure and attributes need to be jointly considered for modeling the complex temporal patterns. Most existing work on static graph are not applicable on evolutionary attributed graph.

Learning node latent representations of evolutionary attributed graph is greatly challenging because of its highly time-varying graph structure and node attributes. Some recent efforts have been made on modeling dynamic graphs [10–12, 15, 18]. But these methods have two main limitations: (1) they are not capable of modeling exact temporal patterns in the latent space, and (2) they usually suffer from low efficiency on inferring future graphs due to their stacked recurrent neural network (RNN) design. Effectively capturing the temporal patterns and efficiently inferring future graph can be beneficial for a broad range of real applications such as city traffic forecasting [8], real-time event detection [9], and online user demand prediction [16].

In this work, we propose a novel approach, called Evolutionary Graph Normalizing Flows (EGNF), for representation learning over evolutionary attributed graph. In particular, we leverage the normalizing flows [2, 3, 17], which is a powerful tool for exact density estimation, to learn the complex distribution of node representations in latent space. The proposed framework is composed of three main components: (1) a time-aware graph neural component admits the autoregressive paradigm for aggregating graph information at each time step, (2) an adapted graph recurrent component for updating graph temporal contexts, and (3) a conditional normalizing flows component for capturing the evolution of node representations in latent space along time. Specifically, the third component can be further decomposed into two sub-models of normalizing flows. One is used to capture the distribution of node representations of arbitrary complexity by considering graph temporal contexts as conditions. It learns invertible transformations to map node representations into simple priors conditioning on temporal contexts. The other one is dedicated to capture the evolutionary patterns of prior distributions. During the stage of inference, the proposed model samples and maps conditional prior into node representations by directly applying the inverse of the learned normalizing flows component. So, the proposed model can be greatly efficient for inferring the structure of future unseen graph.

2 PROBLEM DEFINITION

2.1 Research problem

A static graph is denoted as $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of N nodes, i.e., $|\mathcal{V}| = N$, and $\mathcal{E}$ denotes the set of M edges , i.e., $|\mathcal{E}| = M$. The weighted adjacency matrix of G can be denoted as $\mathbf{A} \in \mathbb{R}^{N\times N}$ where each entry $A_{u,v} \in [0, 1]$ represents the strength of an edge $e_{u,v} \in \mathcal{E}$ between a pair of nodes $u, v \in \mathcal{V}$. And, its attribute matrix is denoted as $\mathbf{X} \in \mathbb{R}^{N\times D_r}$ where each row $\mathbf{x}_v$ contains the D_r-dim raw features of node v. A evolutionary attributed graph $\mathcal{G}$ across a set of discrete time steps is defined as a series of observed static graph snapshots, i.e., $\mathcal{G} := \{(G^1, \mathbf{X}^1), \ldots, (G^T, \mathbf{X}^T)\}$ where T is the number of time steps. Each single snapshot $(G^t, \mathbf{X}^t)$ for $t = 1, \ldots, T$ represents an intermediate state of $\mathcal{G}$. So, the dynamics of evolutionary attributed graph $\mathcal{G}$ along time can be revealed from two aspects: (1) the change of graph structure $\{G^t\}_{t=1}^T$, and (2) the change of node attributes $\{\mathbf{X}^t\}_{t=1}^T$. For brevity, we use $\mathcal{V}$ to denote all unique nodes, i.e., $\mathcal{V} = \bigcup_{t=1}^T \mathcal{V}^t$.

Given such a evolutionary attributed graph $\mathcal{G} = \{(G^t, \mathbf{X}^t)\}_{t=1}^T$, we aim to learn a function $dyn(\mathcal{G}) : \mathcal{V} \times \{1, \ldots, T\} \to \mathbb{R}^{D_h}$ that

DLG '20, August 14–18, 2020, Virtual Event, Singapore

ACM ISBN 978-1-4503-9999-9/21/08…$15.00
https://doi.org/10.1145/1122445.1122456

maps each graph node $v \in \mathcal{V}$ into a D_h-dim (typically $D_h \ll N, M$) latent representation vector $\mathbf{h}_v^t$ at each time step $t = 1, \ldots, T$. For any non-trivial evolutionary graph with $T > 1$, the learned node representations $\mathbf{H}^t := [\mathbf{h}_1^t, \ldots, \mathbf{h}_N^t]^\top \in \mathbb{R}^{N \times D_h}$ should preserve both the information of current graph snapshot $(G^t, \mathbf{X}^t)$ and the temporal patterns of $\mathcal{G}$ up to time step t. Specifically, we are interested learning $\mathbf{H}^t$ that can be characterized by (1) revealing the historical evolutionary trend of $\mathcal{G}$ in previous snapshots $(G^1, \mathbf{X}^1), \ldots, (G^{t-1}, \mathbf{X}^{t-1})$; and, (2) being highly indicative about the developing evolutions of $\mathcal{G}$ like link connections in the future.

2.2 Background

2.2.1 Static GNNs. A static graph can be modeled by a static GNN in the form of $stc(G, \mathbf{X}) : \mathcal{V} \to \mathbb{R}^{D_h}$ which maps each node $v \in \mathcal{V}$ into a D_h-dim embedding $\mathbf{h}_v$. The learned $\mathbf{H} \in \mathbb{R}^{N \times D_h}$ fuses the graph structure G with node attributes information $\mathbf{X}$. Spectral method GCN [7] define stc as:

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\, \mathbf{H}^{(l)}\, \mathbf{W}^{(l)}), \tag{1}$$

where $\hat{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, $\mathbf{D}$ is degree matrix of G, $\mathbf{I}$ is identity matrix, $\mathbf{W}^{(l)}$ contains model parameters of the l-th layer. This can be seen as the relaxed spectral formulation of graph convolution operator $\star_G$. To avoid notation conflicts, we use superscript (l) in parentheses to denote the depth of deep model in the following presentation. Spatial methods GraphSAGE [6] and GAT [14] realize stc under the message passing framework [4]:

$$\mathbf{m}_{\mathcal{N}(v)}^{(l+1)} = agg^{(l)}\left(\left\{msg^{(l)}\left(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}, A_{u,v}\right)\right\}_{\forall u \in \mathcal{N}(v)}\right), \tag{2}$$

$$\mathbf{h}_v^{(l+1)} = upd^{(l)}\left(\mathbf{h}_v^{(l)}, \mathbf{m}_{\mathcal{N}(v)}^{(l+1)}\right), \tag{3}$$

where the message function msg first generates contexts from neighbor nodes $\mathcal{N}(v)$, the aggregation function agg merges them into a neighbor message embedding $\mathbf{m}_{\mathcal{N}(v)}$, and update function upd then incorporates $\mathbf{m}_{\mathcal{N}(v)}$ into the node's new representation. And directly applying stc when $\mathcal{G}$ is actively evolving would fail to capture the temporal patterns.

2.2.2 Normalizing flows. It provides a general mechanism for constructing complex probability distributions over continuous random variables. Let $Z \in \mathbb{R}^{D_h}$ be a random variable with a known probability density function $p_Z : \mathbb{R}^{D_h} \to \mathbb{R}$, and let $H \in \mathbb{R}^{D_h}$ be another random variable with complex probability distribution of interest. The main idea is to express $\mathbf{h} \sim p_H(\mathbf{h})$ as an invertible and differentiable transformation g of $\mathbf{z} \sim p_Z(\mathbf{z})$, i.e., $\mathbf{h} = g(\mathbf{z})$. Using the *change of variables* formula, the density of H can be obtained by:

$$p_H(\mathbf{h}) = p_Z(\mathbf{z}) \left|\det J_g(\mathbf{z})\right|^{-1} = p_Z(f(\mathbf{h})) \left|\det J_f(\mathbf{h})\right|, \tag{4}$$

where $J_g(\mathbf{z}) \in \mathbb{R}^{D_h \times D_h}$ is the Jacobian of g evaluated at $\mathbf{z}$, i.e., $J_g(\mathbf{z}) = \frac{\partial g}{\partial \mathbf{z}}$, f is the inverse of g, i.e., $\mathbf{z} = f(\mathbf{h}) = g^{-1}(\mathbf{h})$, $J_f(\mathbf{h}) = \frac{\partial f}{\partial \mathbf{h}}$ is the Jacobian of f at $\mathbf{h}$. Intuitively, one can image the g as expanding/contracting the space in order to fit the density p_Z into p_H. And the the new density p_H is called a pushforward of density p_Z by the function g. The absolute Jacobian determinant $\left|\det J_g(\mathbf{z})\right|$ quantities the relative volume change in a small area at $\mathbf{z}$ due to g.

Constructing an arbitrarily complicated non-linear invertible function g can be difficult. One principle approach to achieve this is exploiting the property that composition of invertible and differentiable functions is itself invertible. Specifically, let $g^{(1)}, \ldots, g^{(L)}$ be a set of invertible and differentiable functions, it can be shown that the composition function $g = g^{(L)} \circ \cdots \circ g^{(1)}$ is also invertible and differentiable. The inverse and Jacobian determinant of g are:

$$g^{-1} = \left(g^{(L)} \circ \cdots \circ g^{(1)}\right)^{-1} = f^{(1)} \circ \cdots \circ f^{(L)} = f, \tag{5}$$

$$\det J_g(\mathbf{z}) = \prod_{l=1}^{L} \det J_{g^{(l)}}\left(\mathbf{z}^{(l)}\right), \tag{6}$$

where $J_{g^{(l)}}\left(\mathbf{z}^{(l)}\right) = \frac{\partial g^{(l)}}{\partial \mathbf{z}^{(l)}}$ is the Jacobian of $g^{(l)}$ and $\mathbf{z}^{(l)} = g^{(l)} \circ \cdots \circ g^{(1)}(\mathbf{z}) = f^{(l+1)} \circ \cdots \circ f^{(L)}(\mathbf{h})$ is the value of the l-th intermediate flow. In this way, a set of bijective functions can be composed to construct successively more expressive functions. For brevity, we drop superscript and use g and f to denote composite normalizing flows models that are adequately expressive.

3 PROPOSED FRAMEWORK

In this section, we present a novel approach Evolutionary Graph Normalizing Flows (EGNF) for representation learning over evolutionary attributed graph. The framework is illustrated in Figure 1.

3.1 Time-aware graph neural networks

For uncovering the temporal patterns of evolutionary attributed graph $\mathcal{G}$, its intermediate graph structure and node attributes at each time step needs to be modeled and transformed into an unified latent space beforehand. Most existing methods simply apply off-the-shelf GNNs on each snapshot $\left\{(G^t, \mathbf{X}^t)\right\}_{t=1}^{T}$ with sharing parameters. This leads to independent snapshot aggregations across time which ignore the informative context of previous changes of graph. To facilitate enabling time-aware aggregations on each graph snapshot, it is of great importance to make the base GNN model be aware of the temporal contexts of evolutionary attributed graph.

Let's suppose we have already learned $\mathbf{C}^{t-1} \in \mathbb{R}^{N \times D_c}$ summarizing the temporal contexts of $\mathcal{G}$. Each row $\mathbf{c}_v^{t-1}$ reflects the local changing trend associated with node v before time t. We propose a time-aware graph neural module which aggregates graph structural and attributes information from snapshot $(G^t, \mathbf{X}^t)$ with temporal contexts $\mathbf{C}^{t-1}$ as conditional variable. This can be achieved by injecting $\mathbf{C}^{t-1}$ into the aggregated neighbor message (see Eqn. (2)) and the node representation updating process (see Eqn. (3)). Particularly, we add temporal contexts of neighbor nodes as input into the message function msg, and add the target node's temporal context into the representation update function upd:

$$\begin{aligned} \mathbf{m}_u^{t,(l+1)} &= msg^{(l)}\left(\mathbf{h}_v^{t,(l)}, \mathbf{h}_u^{t,(l)}, A_{u,v}^t, \mathbf{c}_u^{t-1}\right) \\ &= A_{u,v}^t \left(\mathbf{h}_v^{t,(l)} + \mathrm{FNN}_m^{(l)}\left(\left[\mathbf{h}_u^{t,(l)}; \mathbf{c}_u^{t-1}\right]\right)\right), \end{aligned} \tag{7}$$

$$\begin{aligned} \mathbf{h}_v^{t,(l+1)} &= upd^{(l)}\left(\mathbf{h}_v^{t,(l)}, \mathbf{m}_{\mathcal{N}(v)}^{t,(l+1)}, \mathbf{c}_v^{t-1}\right) \\ &= \mathrm{FNN}_u^{(l)}\left[\mathbf{h}_v^{t,(l)}; \mathbf{m}_{\mathcal{N}(v)}^{t,(l+1)}; \mathbf{c}_v^{t-1}\right], \end{aligned} \tag{8}$$

where $\mathbf{c}_u^{t-1}$ is the temporal context of neighbor node $u \in \mathcal{N}(v)$, $\mathrm{FNN}_m^{(l)}$ is arbitrary deep model for fusing u's embedding $\mathbf{h}_u^{t,(l)}$ with $\mathbf{c}_u^{t-1}$ (with adding self's embedding $\mathbf{h}_v^{t,(l)}$ as skip connection), $\mathbf{c}_v^{t-1}$ is the temporal context of target node v, and $\mathrm{FNN}_u^{(l)}$ is arbitrary deep

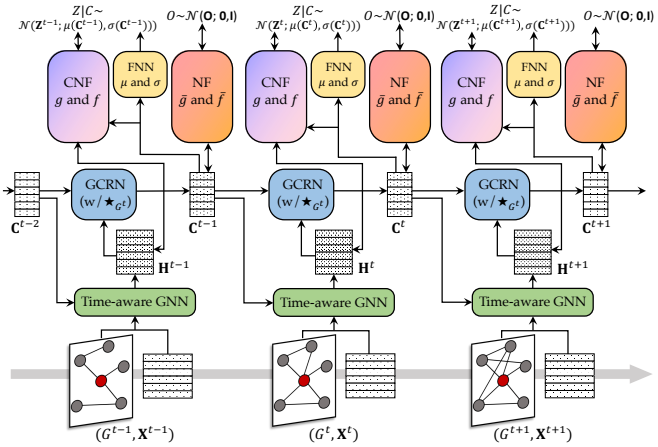

Figure 1: Overall framework of the proposed EGNF.

model for fusing v's old embedding $\mathbf{h}_v^{t,(l)}$ and neighbors' message $\mathbf{m}_{\mathcal{N}(v)}^{t,(l+1)}$ with $\mathbf{c}_v^{t-1}$. In practice, we implement $\text{FNN}_m^{(l)}$ and $\text{FNN}_u^{(l)}$ as one-layer feedforward network with the sigmoid nonlinearity. This time-aware GNN module $\mathbf{H}^t = stc'\left((G^t, \mathbf{X}^t) \,|\, \mathbf{C}^{t-1}\right)$ $(\mathbf{C}^0 = \mathbf{I})$ considers temporal contexts of previous time when aggregating on current snapshot and admit the autoregressive paradigm.

3.2 Graph temporal contexts

For updating the temporal contexts $\mathbf{C}^t$ at each time step, a straightforward way is to employ a deep recurrent architecture. GCRN [13] replaced the multiplications between input and dense parameter matrices in GRU [1] with the standard graph convolution operator $\star_G$ to build a hybrid graph recurrent model. But this method can only handle changes of node attributes because $\star_G$ is fixed by the static graph structure. We propose to generalize the graph recurrent model into truly evolutionary attributed graph setting where the structure and attributes of graph are jointly evolving. In particular, we define a new temporal graph convolution operator $\star_{G^t}$ based on graph structure at t and substitute it into the model:

$$\begin{aligned}
\mathbf{U}^t &= \sigma\left(\mathbf{W}_U \star_{G^t} \left[\mathbf{C}^{t-1}; \mathbf{H}^t\right]\right), \\
\mathbf{R}^t &= \sigma\left(\mathbf{W}_R \star_{G^t} \left[\mathbf{C}^{t-1}; \mathbf{H}^t\right]\right), \\
\widetilde{\mathbf{C}}^t &= \tanh\left(\mathbf{W}_C \star_{G^t} \left[\mathbf{R}^t \odot \mathbf{C}^{t-1}; \mathbf{H}^t\right]\right), \\
\mathbf{C}^t &= \mathbf{U}^t \odot \mathbf{C}^{t-1} + (\mathbf{1} - \mathbf{U}^t) \odot \widetilde{\mathbf{C}}^t,
\end{aligned} \tag{9}$$

where $\mathbf{U}^t \in \mathbb{R}^{N \times D_c}$ is the update gate, $\mathbf{R}^t \in \mathbb{R}^{N \times D_c}$ is the reset gate, and $\mathbf{W}_U, \mathbf{W}_R, \mathbf{W}_C \in \mathbb{R}^{D_c \times D_h}$ are model parameters.

3.3 Graph conditional normalizing flows

We have obtained node latent representations $\mathbf{H}^t$ via time-aware aggregations on the current snapshot $(G^t, \mathbf{X}^t)$. We have also updated the graph temporal contexts $\mathbf{C}^t$ up to time t. For effectively modeling the temporal patterns of evolutionary attributed graph $\mathcal{G}$, we propose a novel deep architecture leveraging conditional normalizing flows [17] to learn invertible transformations between the distribution of node latent representations and a prescribed simple prior distribution. In particular, we introduce the graph temporal context as a conditional variable, and transform complex conditional distribution of node representations $p_{H|C}(\mathbf{H}^t|\mathbf{C}^t)$ into a simple conditional prior $p_{Z|C}(\mathbf{Z}^t|\mathbf{C}^t)$. Specifically, the temporal conditioning is realized from two aspects. On one hand, we formulate the prior distribution as a diagonal Gaussian with its parameters conditioning on the temporal contexts $\mathbf{C}^t$ of graph:

$$p_{Z|C}\left(\mathbf{Z}^t|\mathbf{C}^t\right) = \mathcal{N}\left(\mathbf{Z}^t \,|\, \mu(\mathbf{C}^t), \sigma(\mathbf{C}^t)\right), \tag{10}$$

where μ and σ are deep models for inferring the mean and variance parameters of $Z|C$. Intuitively, conditioning Z on temporal contexts C allows the prior to reflect the general trend of graph changes.

On the other hand, we also impose conditional constraint on the invertible transformation of normalizing flows $g = f^{-1}$ which are bijective between Z and H, i.e., $\mathbf{H}^t = g\left(\mathbf{Z}^t, \mathbf{C}^t\right) = g\left(f\left(\mathbf{H}^t, \mathbf{C}^t\right), \mathbf{C}^t\right)$. In practice, we choose to implement g and f using coupling flows model [3] because of its simplicity. We pass $\mathbf{C}^t$ to scale and translation functions. So, the conditional coupling of g is specified as:

$$\begin{cases}
\mathbf{H}^t_{\langle 1:d-1\rangle} = & \mathbf{Z}^t_{\langle 1:d-1\rangle} \odot \exp\left(c_1\left(\left[\mathbf{Z}^t_{\langle d:D_h\rangle}; \mathbf{C}^t\right]\right)\right) + \\
& c_2\left(\left[\mathbf{Z}^t_{\langle d:D_h\rangle}; \mathbf{C}^t\right]\right), \\
\mathbf{H}^t_{\langle d:D_h\rangle} = & \mathbf{Z}^t_{\langle d:D_h\rangle},
\end{cases} \tag{11}$$

where $\langle 1 : d-1\rangle$ denotes the operator of column-wise partition, e.g., $\mathbf{H}^t = [\mathbf{H}^t_{\langle 1:d-1\rangle}; \mathbf{H}^t_{\langle d:D_h\rangle}]$; $c_1, c_2 : \mathbb{R}^{D_h+D_c} \to \mathbb{R}^{D_h}$ are scaling and translation functions of the coupling layer (implemented as single layer FNN with the sigmoid nonlinearity). And, the inverse function f can be easily obtained as:

$$\begin{cases}
\mathbf{Z}^t_{\langle 1:d-1\rangle} = & \left(\mathbf{H}^t_{\langle 1:d-1\rangle} - c_2\left(\left[\mathbf{Z}^t_{\langle d:D_h\rangle}; \mathbf{C}^t\right]\right)\right) \odot \\
& \exp\left(-c_1\left(\left[\mathbf{Z}^t_{\langle d:D_h\rangle}; \mathbf{C}^t\right]\right)\right), \\
\mathbf{Z}^t_{\langle d:D_h\rangle} = & \mathbf{H}^t_{\langle d:D_h\rangle},
\end{cases} \tag{12}$$

where the Jacobian determinant of f is $\exp(-\sum_i c_1([\mathbf{Z}^t_{\langle d:D_h\rangle}; \mathbf{C}^t])_i)$. We alternate the partition of dimensions $\langle 1 : d-1\rangle$ and $\langle d : D_h\rangle$ between two consecutive conditional coupling layers to ensure all elements in g and f get sufficiently transformed. So, the conditional density of node latent representations H can be obtained as:

$$\begin{aligned}
p_{H|C}\left(\mathbf{H}^t|\mathbf{C}^t\right) &= p_{Z|C}\left(\mathbf{Z}^t|\mathbf{C}^t\right)\left|\det J_g\left(\mathbf{Z}^t\right)\right|^{-1} \\
&= p_{Z|C}\left(\mathbf{Z}^t|\mathbf{C}^t\right)\left|\det \frac{\partial g\left(\mathbf{Z}^t, \mathbf{C}^t\right)}{\partial \mathbf{Z}^t}\right|^{-1} \\
&= p_{Z|C}\left(f\left(\mathbf{H}^t, \mathbf{C}^t\right) | \mathbf{C}^t\right)\left|\det J_f\left(\mathbf{H}^t\right)\right| \\
&= p_{Z|C}\left(f\left(\mathbf{H}^t, \mathbf{C}^t\right) | \mathbf{C}^t\right)\left|\det \frac{\partial f\left(\mathbf{H}^t, \mathbf{C}^t\right)}{\partial \mathbf{H}^t}\right|.
\end{aligned} \tag{13}$$

Then, by substituting the form of conditional prior (Eqn. (10)), and functions of g and f (Eqn. (11) and (12)) into Eqn. (13), we can conduct exact density estimation on node latent representations.

Another major advantage of normalizing flows lies in its high efficiency during the inference stage. We aim to sample and transform conditional prior $\mathbf{Z}^t$ into node representations $\mathbf{H}^t$ by applying the learned flows g in the generative direction. But both the sampling and generation are conditioned on temporal contexts $\mathbf{C}^t$. To reduce the time complexity of generating $\mathbf{C}^t$ via the graph recurrent architecture (Sec. 3.2), we propose to model the distribution of temporal contexts $\mathbf{C}^t$ also using a normalizing flows model. Specifically, we introduce variable O as an isotropic Gaussian prior,

i.e., $p_O(\mathbf{O}) = \mathcal{N}(\mathbf{O} \mid \mathbf{0}, \mathbf{I})$, and learn the invertible transformations $\mathbf{C}^t = \bar{g}(\mathbf{O}) = \bar{g}\left(\bar{f}(\mathbf{C}^t)\right)$ between $\mathbf{C}^t$ and simple prescribed $\mathbf{O}$:

$$\begin{aligned} p_C(\mathbf{C}^t) &= p_O(\mathbf{O}) \left|\det J_{\bar{g}}(\mathbf{O})\right|^{-1} \\ &= p_O(\bar{f}(\mathbf{C}^t)) \left|\det J_{\bar{f}}(\mathbf{C}^t)\right|, \end{aligned} \tag{14}$$

where $\bar{g}$ and $\bar{f}$ are implemented using standard coupling flows [3]. During the inference time, we first sample and transform priors $\mathbf{O}$ to get graph temporal contexts $\mathbf{C}^t = \bar{g}(\mathbf{O})$; next, we generate the conditional prior $\mathbf{Z}^t|\mathbf{C}^t$, then transform into node representations $\mathbf{H}^t = g\left(\mathbf{Z}^t, \mathbf{C}^t\right)$ for predictions on evolutionary attributed graph.

3.4 Complexity Analysis

Assuming the per-batch time complexity of the proposed EGNF's time-aware graph neural component *stc'* (Sec. 3.1) is $O\left(\Pi_{l=1}^{L} s_l\right)$ in principle [6] (where L is the structural depth and s_l is the neighbor sampling size at the l-th layer), and the per-batch time complexity of EGNF's adapted graph recurrent component (Sec. 3.2) at each time step takes constant time. The per-batch time complexity of EGNF during training stage is $O\left(T \Pi_{l=1}^{L} s_l\right)$, which increases linearly with the product of time steps T and neighbor sampling size s_l. At inference time, assuming the transformation of any normalizing flows takes constant time, the time complexity of EGNF for inferring the structure of future graph is $O(N)$, which only grows linearly with number of nodes in evolutionary attributed graph $\mathcal{G}$.

4 EXPERIMENTS

In this section, we evaluate the effectiveness of EGNF. We test on predicting the structure of *next* future graph in all experiments.

4.1 Datasets

Evolutionary co-authorship graphs. We built a sequence of yearly co-authorship graphs by collecting 226,611 papers from 2001 to 2010 in computer science from the Microsoft Academic Graph. Authors were ranked by their number of papers. The top 2,000 and 10,000 were used to make two datasets denoted by $\mathcal{G}_{\text{AU}}^{\text{2K}}$ and $\mathcal{G}_{\text{AU}}^{\text{10K}}$. The venues and the paper title's words were used as node attributes after filtering out infrequent ones. We have 316 venues and 3,549 words in $\mathcal{G}_{\text{AU}}^{\text{2K}}$; and 448 venues, 6,442 words in $\mathcal{G}_{\text{AU}}^{\text{10K}}$.

Evolutionary virtual currency graphs. We used 2 benchmark datasets Bitcoin-OTC and Bitcoin-Alpha of Bitcoin transaction networks denoted by $\mathcal{G}_{\text{BC}}^{\text{otc}}$ and $\mathcal{G}_{\text{BC}}^{\text{alp}}$. These are two who-trusts-whom networks of bitcoin users trading on public platforms. Specifically, we followed the treatments as provided in [11] to form a sequence of graphs with 137 time steps (each for about 2 weeks), and use node in/out degree as input features.

4.2 Experimental settings

Baselines. We compare EGNF against the state-of-the-art methods for modeling evolutionary attributed graph:

- DySAT [12]: This dynamic network embedding method only models graph structural changes and cannot handle node attributes. All graph structures are used for training.
- DCRNN [8]: The most recent graph structure and all node attribute matrices are used for training. The node embeddings outputted by the last diffusion convolutional layer are used for predicting future graph structure.
- EvolveGCN [11]: All graph snapshots are provided as input. We use its link prediction loss for training, and use the node embeddings outputted by the last evolving graph convolution unit for predicting future graph structure.
- VGRNN [5]: All graph snapshots are provided as input for training. We use the advanced model variant with semi-implicit hierarchical construction of mixing prior distribution. The learned probabilistic node representations are used for predicting future graph structure.

Evaluation metrics. For link prediction on future graph, we use Area Under the precision-recall Curve (AUC) and F1 measure.

4.3 Overall performance (RQ1)

The performance of the proposed EGNF and baselines for predicting future graph structure are provided in Table 1. We can see the proposed EGNF can almost outperform all baseline methods across datasets and metrics (except for AUC on $\mathcal{G}_{\text{AU}}^{\text{10K}}$ and $\mathcal{G}_{\text{BC}}^{\text{otc}}$). By leveraging normalizing flows to preserve the exact temporal patterns of the evolutionary attributed graph, EGNF can score an AUC of .151 on $\mathcal{G}_{\text{AU}}^{\text{2K}}$ (+4.1% relatively over VGRNN) and score and F1 of .441 on $\mathcal{G}_{\text{BC}}^{\text{alp}}$ (+4.0% relatively over EvolveGCN). The significant performance improvements brought by EGNF over competitive baseline methods validate the contributions of modeling and preserving the exact temporal patterns of evolutionary attributed graph via normalizing flows over time. In addition, we note that EGNF can produce stable performance for predicting the future graph structure. It yields relatively small values of variation across datasets and metrics (only slightly larger than EvolveGCN). Although the inference stage of EGNF includes random sampling over (conditional) priors, the learned invertible transformations of normalizing flows can effectively map sampled priors into informative node representations for predicting the future graph structure.

4.4 Ablation study (RQ2)

We build four model variants by disabling/removing certain component(s) out, and compare EGNF against these variants:

- (base): All three components are removed from EGNF (and adopt a single normalizing flows model for learning node representations). This variant serves as the baseline variant for verifying the contribution of each component.
- (w/o tGNN): We remove $\mathbf{C}^{t-1}$ from the *msg* and *upd* functions of the proposed time-aware graph neural networks component (see Sec. 3.1). This equals to perform independent aggregations on graph snapshot at each time step
- (w/o tGCRN): We replace the proposed temporal graph convolution operator $\star_{G^t}$ with the conventional one being fixed at the first graph structure G^1 in the proposed graph recurrent component (see Sec. 3.2). This equals to using a fixed graph structure for updating temporal contexts.
- (w/o cNF): We remove the conditioning on temporal context $\mathbf{C}^t$ from g and f for estimating node representations, and disable invertible transformations $\bar{g}$ and $\bar{f}$ for estimating $\mathbf{C}^t$

Table 1: Overall performance of the proposed EGNF and baseline methods for predicting the graph structure at the last time step on four real evolutionary attributed graph data sets. Bold and underlined values indicate the best and second performance.

Method	$\mathcal{G}_{\text{AU}}^{\text{2K}}$		$\mathcal{G}_{\text{AU}}^{\text{10K}}$		$\mathcal{G}_{\text{BC}}^{\text{otc}}$		$\mathcal{G}_{\text{BC}}^{\text{alp}}$	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
DySAT [12]	.117 ± .008	.212 ± .007	.112 ± .011	.203 ± .012	.287 ± .007	.290 ± .006	.312 ± .006	.331 ± .006
DCRNN [8]	.123 ± .003	.224 ± .003	.121 ± .004	.218 ± .005	.312 ± .002	.338 ± .001	.340 ± .002	.352 ± .002
EvolveGCN [11]	.133 ± .001	.256 ± .001	.127 ± .001	.249 ± .002	**.414** ± .002	.420 ± .001	.415 ± .001	.424 ± .001
VGRNN [5]	.145 ± .003	.266 ± .005	**.144** ± .003	.262 ± .004	.354 ± .004	.385 ± .005	.362 ± .003	.382 ± .004
EGNF	**.151** ± .002	**.275** ± .002	.143 ± .002	**.269** ± .002	.412 ± .002	**.428** ± .003	**.426** ± .002	**.441** ± .002

Table 2: Performance of the proposed EGNF and its variants for predicting the graph structure at the last time step on four real evolutionary attributed graph data sets. Relative improvements over the base variant are shown in parenthesis.

Method	$\mathcal{G}_{\text{AU}}^{\text{2K}}$		$\mathcal{G}_{\text{AU}}^{\text{10K}}$		$\mathcal{G}_{\text{BC}}^{\text{otc}}$		$\mathcal{G}_{\text{BC}}^{\text{alp}}$	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
EGNF (base)	.141 ± .001	.262 ± .001	.138 ± .001	.257 ± .002	.378 ± .001	.396 ± .002	.389 ± .001	.407 ± .001
EGNF (w/o tGNN)	.148 ± .002 (+5.0%)	.270 ± .001 (+3.1%)	**.143** ± .001 (+**3.6**%)	.268 ± .001 (+4.3%)	.386 ± .001 (+2.1%)	.407 ± .003 (+2.8%)	.409 ± .002 (+5.1%)	.429 ± .001 (+5.4%)
EGNF (w/o tGCRN)	.150 ± .001 (+6.4%)	.273 ± .002 (+4.2%)	.141 ± .002 (+2.2%)	.266 ± .002 (+3.5%)	.406 ± .001 (+7.4%)	.421 ± .002 (+6.3%)	.417 ± .001 (+7.2%)	.436 ± .001 (+7.1%)
EGNF (w/o cNF)	.144 ± .001 (+2.1%)	.267 ± .002 (+1.9%)	.138 ± .002 (+0.0%)	.258 ± .002 (+0.1%)	.378 ± .003 (+0.0%)	.398 ± .002 (+0.1%)	.398 ± .002 (+2.3%)	.417 ± .001 (+2.5%)
EGNF (full)	**.151** ± .002 (+**7.1**%)	**.275** ± .002 (+**5.0**%)	**.143** ± .002 (+**3.6**%)	**.269** ± .002 (+**4.7**%)	**.412** ± .002 (+**9.0**%)	**.428** ± .003 (+**8.1**%)	**.426** ± .002 (+**9.4**%)	**.441** ± .002 (+**8.4**%)

(Sec. 3.3). This equals using a standard normalizing flows model to capture the temporal patterns.

We present the results of ablation study in Table 2. WE can see the variant EGNF (w/o cNF) generally yield the lowest values of performance improvement over EGNF (base). We note it almost performs the same as EGNF (base) on $\mathcal{G}_{\text{AU}}^{\text{10K}}$ and $\mathcal{G}_{\text{BC}}^{\text{otc}}$. This demonstrates modeling the distribution of node representations using conditional normalizing flows is critic for predicting the future graph structure. In contrast, by fully including the conditional normalizing flows component for preserving graph temporal patterns, the proposed EGNF consistently outperforms all variants across datasets and metrics. It scores an AUC of .412 on $\mathcal{G}_{\text{BC}}^{\text{otc}}$ which is +9.0% relatively over EGNF (base) and EGNF (w/o cNF). And, it scores an F1 of .441 $\mathcal{G}_{\text{BC}}^{\text{alp}}$ which is +8.4% relatively over EGNF (base) (and +5.8% relatively over EGNF (w/o cNF)). We conclude that all three components of EGNF are important and can be complementary to model the temporal patterns of evolutionary attributed graph.

5 CONCLUSIONS

In this work, we proposed a novel approach for representation learning over evolutionary attributed graph utilizing conditional normalizing flows. It has three components: (1) a time-aware graph neural component for aggregating graph snapshot at each time step, (2) an adapted graph recurrent component for updating graph temporal contexts, and (3) a conditional normalizing flows component for capturing the evolution of node representations in latent space along time. The proposed approach is efficient in inference by sampling and mapping conditional prior into node representations using the inverse of learned flows component.

REFERENCES

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

[2] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

[3] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803* (2016).

[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.

[5] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Duffield Narayanan, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational Graph Recurrent Neural Networks. *NeurIPS* (2019).

[6] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st NeurIPS*.

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

[8] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

[9] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. 2020. Dynamic Heterogeneous Graph Neural Network for Real-time Event Prediction. In *Proceedings of the 26th ACM SIGKDD*.

[10] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.

[11] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*.

[12] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th WSDM*.

[13] 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*. Springer.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[15] Daheng Wang, Zhihan Zhang, Yihong Ma, Tong Zhao, Tianwen Jiang, Nitesh V Chawla, and Meng Jiang. 2020. Learning Attribute-Structure Co-Evolutions in Dynamic Graphs. *arXiv preprint arXiv:2007.13004* (2020).

[16] Qianru Wang, Bin Guo, Yi Ouyang, Kai Shu, Zhiwen Yu, and Huan Liu. 2020. Spatial Community-Informed Evolving Graphs for Demand Prediction. In *Proceedings of ECMLPKDD 2020*.

[17] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. 2019. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042* (2019).

[18] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *AAAI*.