Variational Graph Recurrent Neural Networks

Ehsan Hajiramezanali[†*], Arman Hasanzadeh[†*], Nick Duffield[†], Krishna Narayanan[†],
Mingyuan Zhou[‡], Xiaoning Qian[†]

† Department of Electrical and Computer Engineering, Texas A&M University
{ehsanr, armanihm, duffieldng, krn, xqian}@tamu.edu
‡ McCombs School of Business, University of Texas at Austin
mingyuan.zhou@mccombs.utexas.edu

GRNN VGRNN

Abstract

Representation learning over graph structured data has been mostly studied in static graph settings while efforts for modeling dynamic graphs are still scant. In this paper, we develop a novel hierarchical variational model that introduces additional latent random variables to jointly model the hidden states of a graph recurrent neural network (GRNN) to capture both topology and node attribute changes in dynamic graphs. We argue that the use of high-level latent random variables in this variational GRNN (VGRNN) can better capture potential variability observed in dynamic graphs as well as the uncertainty of node latent representation. With semi-implicit variational inference developed for this new VGRNN architecture (SI-VGRNN), we show that flexible non-Gaussian latent representations can further help dynamic graph analytic tasks. Our experiments with multiple real-world dynamic graph datasets demonstrate that SI-VGRNN and VGRNN consistently outperform the existing baseline and state-of-the-art methods by a significant margin in dynamic link prediction.

1 Introduction

Node embedding maps each node in a graph to a vector in a low-dimensional latent space, in which classical feature vector-based machine learning formulations can be adopted [4]. Most of the existing node embedding techniques assume that the graph is static and that learning tasks are performed on fixed sets of nodes and edges [29, 37, 12, 32, 23, 21]. However, many real-world problems are modeled by *dynamic* graphs, where graphs are constantly evolving over time. Such graphs have been typically observed in social networks, citation networks, and financial transaction networks. A naive solution to node embedding for dynamic graphs is simply applying static methods to each snapshot of dynamic graphs. Among many potential problems of such a naive solution, it is clear that it ignores the temporal dependencies between snapshots.

Several node embedding methods have been proposed to capture the temporal graph evolution for both networks without attributes [10, 44] and attributed networks [39, 26]. However, all of the existing dynamic graph embedding approaches represent each node by a deterministic vector in a low-dimensional space [1]. Such deterministic representations lack the capability of modeling uncertainty of node embedding, which is a natural consideration when having multiple information sources, i.e. node attributes and graph structure.

In this paper, we propose a novel node embedding method for dynamic graphs that maps each node to a random vector in the latent space. More specifically, we first introduce a dynamic graph autoencoder model, namely graph recurrent neural network (GRNN), by extending the use of graph convolutional neural networks (GCRN) [35] to dynamic graphs. Then, we argue that GRNN lacks the expressive power for fully capturing the complex dependencies between topological evolution and time-varying node attributes because the output probability in standard RNNs is limited to either a simple unimodal distribution or a mixture of unimodal distributions [2, 36, 5, 8]. Next, to increase the expressive power of GRNN in addition to modeling the uncertainty of node latent representations, we propose variational graph recurrent neural network (VGRNN) by adopting high-level latent random variables in GRNN. Our proposed VGRNN is capable of learning interpretable latent representation as well as better modeling of very sparse dynamic graphs.

To further boost the expressive power and interpretability of our new VGRNN method, we integrate semi-implicit variational inference [42] with VGRNN. We show that semi-implicit variational graph recurrent neural network (SI-VGRNN) is capable of inferring more flexible and complex posteriors. Our experiments demonstrate the superior performance of VGRNN and SI-VGRNN in dynamic link prediction tasks in several real-world dynamic graph datasets compared to baseline and state-of-the-art methods.

2 Background

Graph convolutional recurrent networks (GCRN). GCRN was introduced by Seo et al. [35] to model time series data defined over nodes of a static graph. Series of frames in videos and spatio-temporal measurements on a network of sensors are two examples of such datasets. GCRN combines graph convolutional networks (GCN) [3] with recurrent neural networks (RNN) to capture spatial and temporal patterns in data. More precisely, given a graph G with N nodes, whose topology is determined by the adjacency matrix

---

*Both authors contributed equally.

$\mathbf{A} \in \mathbb{R}^{N \times N}$, and a sequence of node attributes $\mathcal{X} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(T)}\}$, GCRN reads M-dimensional node attributes $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times M}$ and updates its hidden state $\mathbf{h}_t \in \mathbb{R}^p$ at each time step t:

$$\mathbf{h}_t = f\left(\mathbf{A}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}\right). \tag{1}$$

Here f is a non-probabilistic deep neural network. It can be any recursive network including gated activation functions such as long short-term memory (LSTM) or gated recurrent units (GRU), where the deep layers inside them are replaced by graph convolutional layers. GCRN models node attribute sequences by parameterizing a factorization of the joint probability distribution as a product of conditional probabilities such that

$$p\left(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(T)} \mid \mathbf{A}\right) = \prod_{t=1}^{T} p\left(\mathbf{X}^{(t)} \mid \mathbf{X}^{(<t)}, \mathbf{A}\right); \quad p\left(\mathbf{X}^{(t)} \mid \mathbf{X}^{(<t)}, \mathbf{A}\right) = g(\mathbf{A}, \mathbf{h}_{t-1}).$$

Due to the deterministic nature of the transition function f, the choice of the mapping function g here effectively defines the only source of variability in the joint probability distributions $p(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(T)} \mid \mathbf{A})$ that can be expressed by the standard GCRN. This can be problematic for sequences that are highly variable. More specifically, when the variability of X is high, the model tries to map this variability in hidden states h, leading to potentially high variations in h and thereafter overfitting of training data. Therefore, ~~GCRN is not fully capable of modeling sequences with high variations~~. This fundamental problem of autoregressive models has been addressed for non-graph-structured datasets by introducing stochastic hidden states to the model [2, 6, 9, 13, 15, 16, 18, 17].

In this paper, we integrate GCN and RNN into a graph RNN (GRNN) framework, which is a dynamic graph autoencoder model. While GCRN aims to model dynamic node attributes defined over a static graph, GRNN can get different adjacency matrices at different time snapshots and reconstruct the graph at time t by adopting an inner-product decoder on the hidden state $\mathbf{h}_t$. More specifically, $\mathbf{h}_t$ can be viewed as node embedding of the dynamic graph at time t. To further improve the expressive power of GRNN, we introduce stochastic latent variables by combining GRNN with variational graph autoencoder (VGAE) [23]. This way, not only we can capture time dependencies between graphs without making smoothness assumption, but also each node is represented with a distribution in the latent space. Moreover, the prior construction devised in VGRNN allows it to predict links in the future time steps.

Semi-implicit variational inference (SIVI). SIVI has been shown effective to learn posterior distributions with skewness, kurtosis, multimodality, and other characteristics, which were not captured by the existing variational inference methods [42]. To characterize the latent posterior $q(\mathbf{z}|\mathbf{x})$, SIVI introduces a mixing distribution on the parameters of the original posterior distribution to expand the variational family with a hierarchical construction: $\mathbf{z} \sim q(\mathbf{z}|\boldsymbol{\psi})$ with $\boldsymbol{\psi} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi})$. $\boldsymbol{\phi}$ denotes the distribution parameter to be inferred. While the original posterior $q(\mathbf{z}|\boldsymbol{\psi})$ is required to have an analytic form, its mixing distribution is not subject to such a constraint, and so the marginal posterior distribution is often implicit and more expressive that has no analytic density function. It is also common that the marginal of the hierarchy is implicit, even if both the posterior and its mixing distribution are explicit. We will integrate SIVI in our new model to infer more flexible and interpretable node embedding for dynamic graphs.

3 Variational graph recurrent neural network (VGRNN)

3.1 Overview

We consider a dynamic graph $\mathcal{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$ where $G^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ is the graph at time step t with $\mathcal{V}^{(t)}$ and $\mathcal{E}^{(t)}$ being the corresponding node and edge sets, respectively. In this paper, we aim to develop a model that is universally compatible with potential changes in both node and edge sets. In particular, the cardinality of both $\mathcal{V}^{(t)}$ and $\mathcal{E}^{(t)}$ can change across time. There are no constraints on the relationships between $(\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ and $(\mathcal{V}^{(t+1)}, \mathcal{E}^{(t+1)})$, namely new nodes can join the dynamic graph and create edges to the existing nodes or previous nodes can disappear from the graph. On the other hand, new edges can form between snapshots while existing edges can disappear. Therefore, VGRNN can take as input a variable-length adjacency matrix sequence $\mathcal{A} = \{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(T)}\}$. In addition, when considering node attributes, different attributes can be observed at different snapshots with a variable-length node attribute sequence $\mathcal{X} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(T)}\}$. Inspired by variational recurrent neural networks (VRNN) [2], we construct VGRNN by integrating GRNN and VGAE so that complex dependencies between topological and node attribute dynamics are modeled sufficiently and simultaneously. Moreover, each node at each time is represented with a distribution, hence uncertainty of latent representations of nodes are also modelled in VGRNN.

3.2 VGRNN model

Generation. The VGRNN model adopts a VGAE to model each graph snapshot. The VGAEs across time are conditioned on the state variable $\mathbf{h}_{t-1}$, modeled by a GRNN. Such an architecture design will help each VGAE to take into account the temporal structure of the dynamic graph. More critically, unlike a standard VGAE, Our VGAE in VGRNN takes a new prior on the latent random variables by allowing distribution parameters to be modelled by either explicit or implicit complex functions of information of the previous time step. More specifically, instead of imposing a standard multivariate Gaussian distribution with deterministic parameters, VGAE in our VGRNN learns the prior distribution parameters based on the hidden states in previous time steps. Hence, our VGRNN allows more flexible latent representations with greater expressive power that captures dependencies between and within topological and node attribute evolution processes. In

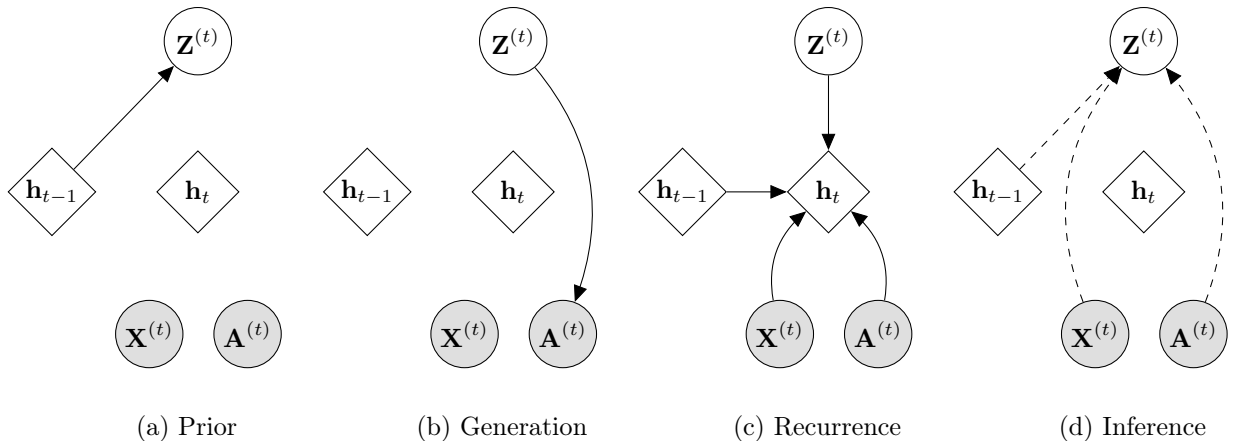| (a) Prior | (b) Generation | (c) Recurrence | (d) Inference |

Figure 1: Graphical illustrations of each operation of VGRNN; (a) computing the conditional prior by (2); (b) decoder function (3); (c) updating the GRNN hidden states using (4); and (d) inference of the posterior distribution for latent variables by (3.2).

particular, we can write the construction of the prior distribution adopted in our experiments as follows,

$$p\left(\mathbf{Z}^{(t)}\right) = \prod_{i=1}^{N} p\left(\mathbf{Z}_i^{(t)}\right); \ \ \mathbf{Z}_i^{(t)} \sim \mathcal{N}\left(\boldsymbol{\mu}_{i,\mathrm{prior}}^{(t)}, \mathrm{diag}((\boldsymbol{\sigma}_{i,\mathrm{prior}}^{(t)})^2)\right), \ \ \left\{\boldsymbol{\mu}_{\mathrm{prior}}^{(t)}, \boldsymbol{\sigma}_{\mathrm{prior}}^{(t)}\right\} = \varphi^{\mathrm{prior}}(\mathbf{h}_{t-1}), \quad (2)$$

where $\boldsymbol{\mu}_{\mathrm{prior}}^{(t)} \in \mathbb{R}^{N \times l}$ and $\boldsymbol{\sigma}_{\mathrm{prior}}^{(t)} \in \mathbb{R}^{N \times l}$ denote the parameters of the conditional prior distribution, and $\boldsymbol{\mu}_{i,\mathrm{prior}}^{(t)}$ and $\boldsymbol{\sigma}_{i,\mathrm{prior}}^{(t)}$ are the i-th row of $\boldsymbol{\mu}_{\mathrm{prior}}^{(t)}$ and $\boldsymbol{\sigma}_{\mathrm{prior}}^{(t)}$, respectively. Moreover, the generating distribution will be conditioned on $\mathbf{Z}^{(t)}$ as:

$$\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)} \sim \mathrm{Bernoulli}\left(\pi^{(t)}\right), \quad \pi^{(t)} = \varphi^{\mathrm{dec}}\left(\mathbf{Z}^{(t)}\right), \quad (3)$$

where $\pi^{(t)}$ denotes the parameter of the generating distribution; φ^{prior} and φ^{dec} can be any highly flexible functions such as neural networks.

On the other hand, the backbone GRNN enables flexible modeling of complex dependency involving both graph topological dynamics and node attribute dynamics. The GRNN updates its hidden states using the recurrence equation:

$$\mathbf{h}_t = f\left(\mathbf{A}^{(t)}, \varphi^{\mathbf{x}}\left(\mathbf{X}^{(t)}\right), \varphi^{\mathbf{z}}\left(\mathbf{Z}^{(t)}\right), \mathbf{h}_{t-1}\right), \quad (4)$$

where f is originally the transition function from equation (1). Unlike the GRNN defined in [35], graph topology can change in different time steps as it does in real-world dynamic graphs, and the adjacency matrix $\mathbf{A}^{(t)}$ is time dependent in VGRNN. To further enhance the expressive power, $\varphi^{\mathbf{x}}$ and $\varphi^{\mathbf{z}}$ are deep neural networks which operate on each node independently and extract features from $\mathbf{X}^{(t)}$ and $\mathbf{Z}^{(t)}$, respectively. These feature extractors are crucial for learning complex graph dynamics. Based on (4), $\mathbf{h}_t$ is a function of $\mathbf{A}^{\leq(t)}$, $\mathbf{X}^{\leq(t)}$, and $\mathbf{Z}^{\leq(t)}$. Therefore, the prior and generating distributions in equations (2) and (3) define the distributions $p(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(<t)}, \mathbf{X}^{(<t)}, \mathbf{Z}^{(<t)})$ and $p(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)})$, respectively. The generative model can be factorized as

$$p\left(\mathbf{A}^{(\leq T)}, \mathbf{Z}^{(\leq T)} \,|\, \mathbf{X}^{(<T)}\right) = \prod_{t=1}^{T} p\left(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(<t)}, \mathbf{X}^{(<t)}, \mathbf{Z}^{(<t)}\right) p\left(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)}\right), \quad (5)$$

where the prior of the first snapshot is considered to be a standard multivariate Gaussian distribution, i.e. $p(\mathbf{Z}_i^{(0)} \,|\, -) \sim \mathcal{N}(0, \mathbf{I})$ for $i \in \{1, \ldots, N_0\}$ and $\mathbf{h}_0 = \mathbf{0}$. Also, if a previously unobserved node is added to the graph at snapshot t, we consider the hidden state of that node at snapshot $t-1$ is zero and hence the prior for that node at time t is $\mathcal{N}(0, \mathbf{I})$.

Inference. With the VGRNN framework, the node embedding for dynamic graphs can be derived by inferring the posterior distribution of $\mathbf{Z}^{(t)}$ which is also a function of $\mathbf{h}_{t-1}$. More specifically,

$$q\left(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}\right) = \prod_{i=1}^{N} q\left(\mathbf{Z}_i^{(t)} \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}\right) = \prod_{i=1}^{N} \mathcal{N}\left(\boldsymbol{\mu}_{i,\mathrm{enc}}^{(t)}, \mathrm{diag}((\boldsymbol{\sigma}_{i,\mathrm{enc}}^{(t)})^2)\right),$$

$$\boldsymbol{\mu}_{\mathrm{enc}}^{(t)} = \mathrm{GNN}_{\mu}\left(\mathbf{A}^{(t)}, \mathrm{CONCAT}\left(\varphi^{\mathbf{x}}\left(\mathbf{X}^{(t)}\right), \mathbf{h}_{t-1}\right)\right),$$

$$\boldsymbol{\sigma}_{\mathrm{enc}}^{(t)} = \mathrm{GNN}_{\sigma}\left(\mathbf{A}^{(t)}, \mathrm{CONCAT}\left(\varphi^{\mathbf{x}}\left(\mathbf{X}^{(t)}\right), \mathbf{h}_{t-1}\right)\right), \quad (6)$$

where $\boldsymbol{\mu}_{\mathrm{enc}}^{(t)}$ and $\boldsymbol{\sigma}_{\mathrm{enc}}^{(t)}$ denote the parameters of the approximated posterior, and $\boldsymbol{\mu}_{i,\mathrm{enc}}^{(t)}$ and $\boldsymbol{\sigma}_{i,\mathrm{enc}}^{(t)}$ are the i-th row of $\boldsymbol{\mu}_{\mathrm{enc}}^{(t)}$ and $\boldsymbol{\sigma}_{\mathrm{enc}}^{(t)}$, respectively. GNN_{μ} and GNN_{σ} are the encoder functions and can be any of the various types of graph neural networks, such as GCN [24], GCN with Chebyshev filters [3] and GraphSAGE [19].

Learning. The objective function of VGRNN is derived from the variational lower bound at each snapshot. More precisely, using equation (5), the evidence lower bound of VGRNN can be written as follows,

$$\mathcal{L} = \sum_{t=1}^{T} \Bigg\{ \mathbb{E}_{\mathbf{Z}^{(t)} \sim q\left(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)}, \mathbf{Z}^{(<t)}\right)} \log p\left(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)}\right)$$

$$- \mathbf{KL}\left(q\left(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)}, \mathbf{Z}^{(<t)}\right) \,||\, p\left(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(<t)}, \mathbf{X}^{(<t)}, \mathbf{Z}^{(<t)}\right)\right) \Bigg\}. \quad (7)$$

Table 1: Dataset statistics.

Metrics	Enron	COLAB	Facebook	HEP-TH	Cora	Social Evolution
Number of Snapshots	11	10	9	40	11	27
Number of Nodes	184	315	663	1199-7623	708-2708	84
Number of Edges	115-266	165-308	844-1068	769-34941	406-5278	303-1172
Average Density	0.01284	0.00514	0.00591	0.00117	0.00154	0.21740
Number of Node Attributes	-	-	-	-	1433	168

We learn the parameters of the generative and inference models jointly by optimizing the variational lower bound with respect to the variational parameters. The graphical representation of VGRNN is illustrated in Fig. 1, operations (a)–(d) correspond to equations (2) – (4), and (3.2), respectively. We note that if we don't use hidden state variables $\mathbf{h}_{t-1}$ in the derivation of the prior distribution, then the prior in (2) becomes independent across snapshots and reduces to the prior of vanilla VGAE.

The inner-product decoder is adopted in VGRNN for the experiments in this paper– φ^{dec} in (3)–to clearly demonstrate the advantages of the stochastic recurrent models for the encoder. Potential extensions with other decoders can be integrated with VGRNN if necessary. More specifically,

$$p\left(\mathbf{A}^{(t)} \mid \mathbf{Z}^{(t)}\right) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left((A_{i,j}^{(t)} \mid \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}\right); p\left(A_{i,j}^{(t)} = 1 \mid \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}\right) = \mathrm{sigmoid}\left(\mathbf{z}_i^{(t)}(\mathbf{z}_j^{(t)})^T\right), \qquad (8)$$

where $\mathbf{z}_i^{(t)}$ corresponds to the embedding representation of node $v_i^{(t)} \in \mathcal{V}^{(t)}$ at time step t. Note the generating distribution can also be conditioned on $\mathbf{h}_{t-1}$ if we want to generate $\mathbf{X}^{(t)}$ in addition to the adjacency matrix for other applications. In such cases, φ^{dec} should be a highly flexible neural network instead of a simple inner-product function.

3.3 Semi-implicit VGRNN (SI-VGRNN)

To further increase the expressive power of the variational posterior of VGRNN, we introduce a SI-VGRNN dynamic node embedding model. We impose a mixing distributions on the variational distribution parameters in (8) to model the posterior of VGRNN with a semi-implicit hierarchical construction:

$$\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t), \qquad \boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)}, \mathbf{Z}^{(<t)}) = q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}). \qquad (9)$$

While the variational distribution $q(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t)$ is required to be explicit, the mixing distribution, $q_{\boldsymbol{\phi}}$, is not subject to such a constraint, leading to considerably flexible $\mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})}(q(\mathbf{z}_t \mid \boldsymbol{\psi}_t))$. More specifically, SI-VGRNN draws samples from $q_{\boldsymbol{\phi}}$ by transforming random noise $\boldsymbol{\epsilon}_t$ via a graph neural network, which generally leads to an implicit distribution for $q_{\boldsymbol{\phi}}$.

Inference. Under the SI-VGRNN construction, the generation, prior and recurrence models are the same as VGRNN (equations (2) to (5)). We indeed have updated the encoder functions as follows:

$$\boldsymbol{\ell}_j^{(t)} = \mathrm{GNN}_j(\mathbf{A}^{(t)}, \mathrm{CONCAT}(\mathbf{h}_{t-1}, \boldsymbol{\epsilon}_j^{(t)}, \boldsymbol{\ell}_{j-1}^{(t)})); \; \boldsymbol{\epsilon}_j^{(t)} \sim q_j(\boldsymbol{\epsilon}) \text{ for } j = 1, \ldots, L, \; \boldsymbol{\ell}_0^{(t)} = \varphi_\tau^{\mathbf{x}}\left(\mathbf{X}^{(t)}\right)$$

$$\boldsymbol{\mu}_{\mathrm{enc}}^{(t)}(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}) = \mathrm{GNN}_\mu(\mathbf{A}^{(t)}, \boldsymbol{\ell}_L^{(t)}), \quad \boldsymbol{\Sigma}_{\mathrm{enc}}^{(t)}(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}) = \mathrm{GNN}_\Sigma(\mathbf{A}^{(t)}, \boldsymbol{\ell}_L^{(t)}),$$

$$q(\mathbf{Z}_i^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}, \boldsymbol{\mu}_{i,\mathrm{enc}}^{(t)}, \boldsymbol{\Sigma}_{i,\mathrm{enc}}^{(t)}) = \mathcal{N}(\boldsymbol{\mu}_{i,\mathrm{enc}}^{(t)}(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}), \boldsymbol{\Sigma}_{i,\mathrm{enc}}^{(t)}(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})),$$

where L is the number of stochastic layers and $\boldsymbol{\epsilon}_j^{(t)}$ is N_t-dimensional random noise drawn from a distribution q_j with N_t denoting number of nodes at time t. Note that given $\{\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}\}$, $\boldsymbol{\mu}_{i,\mathrm{enc}}^{(t)}$ and $\boldsymbol{\Sigma}_{i,\mathrm{enc}}^{(t)}$ are now random variables rather than analytic and thus the posterior is not Gaussian after marginalizing.

Learning. In this construction, because the parameters of the posterior are random variables, the ELBO goes beyond the simple VGRNN in (7) and can be written as

$$\mathcal{L} = \sum_{t=1}^{T} \left\{ \mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbb{E}_{\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t)} \log\left(p(\mathbf{A}^{(t)} \mid \mathbf{Z}^{(t)}, \mathbf{h}_{t-1})\right) \right.$$

$$\left. - \mathbf{KL}\left(\mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} q\left(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t\right) \mid\mid p(\mathbf{Z}^{(t)} \mid \mathbf{h}_{t-1})\right) \right\}. \tag{10}$$

Direct optimization of the ELBO in SIVI is not tractable [42], hence to infer variational parameters of SI-VGRNN, we derive a lower bound for the ELBO as follows (see the supplements for more details.).

$$\underline{\mathcal{L}} = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbb{E}_{\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t)} \log\left(\frac{p(\mathbf{A}^{(t)} \mid \mathbf{Z}^{(t)}, \mathbf{h}_{t-1}) \, p(\mathbf{Z}^{(t)} \mid \mathbf{h}_{t-1})}{q(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t)}\right). \tag{11}$$

4 Experiments

Datasets. We evaluate our proposed methods, VGRNN and SI-VGRNN, and baselines on six real-world dynamic graphs as described in Table 1. More detailed descriptions of the datasets can be found in the supplement.

Competing methods. We compare the performance of our proposed methods against four competing node embedding methods, three of which have the capability to model evolving graphs with changing node

Table 2: AUC and AP scores of inductive dynamic link detection on dynamic graphs.

Metrics	Methods	Enron	COLAB	Facebook	Social Evo.	HEP-TH	Cora
AUC	VGAE	88.26 ± 1.33	70.49 ± 6.46	80.37 ± 0.12	79.85 ± 0.85	79.31 ± 1.97	87.60 ± 0.54
	DynAE	84.06 ± 3.30	66.83 ± 2.62	60.71 ± 1.05	71.41 ± 0.66	63.94 ± 0.18	53.71 ± 0.48
	DynRNN	77.74 ± 5.31	68.01 ± 5.50	69.77 ± 2.01	74.13 ± 1.74	72.39 ± 0.63	76.09 ± 0.97
	DynAERNN	91.71 ± 0.94	77.38 ± 3.84	81.71 ± 1.51	78.67 ± 1.07	82.01 ± 0.49	74.35 ± 0.85
	GRNN	91.09 ± 0.67	86.40 ± 1.48	85.60 ± 0.59	78.27 ± 0.47	89.00 ± 0.46	91.35 ± 0.21
	VGRNN	$\mathbf{94.41 \pm 0.73}$	$\mathbf{88.67 \pm 1.57}$	$\mathbf{88.00 \pm 0.57}$	$\mathbf{82.69 \pm 0.55}$	$\mathbf{91.12 \pm 0.71}$	$\mathbf{92.08 \pm 0.35}$
	SI-VGRNN	$\mathbf{95.03 \pm 1.07}$	$\mathbf{89.15 \pm 1.31}$	$\mathbf{88.12 \pm 0.83}$	$\mathbf{83.36 \pm 0.53}$	91.05 ± 0.92	$\mathbf{94.07 \pm 0.44}$
AP	VGAE	89.95 ± 1.45	73.08 ± 5.70	79.80 ± 0.22	79.41 ± 1.12	81.05 ± 1.53	89.61 ± 0.87
	DynAE	86.30 ± 2.43	67.92 ± 2.43	60.83 ± 0.94	70.18 ± 1.98	63.87 ± 0.21	53.84 ± 0.51
	DynRNN	81.85 ± 4.44	73.12 ± 3.15	70.63 ± 1.75	72.15 ± 2.30	74.12 ± 0.75	76.54 ± 0.66
	DynAERNN	93.16 ± 0.88	83.02 ± 2.59	83.36 ± 1.83	77.41 ± 1.47	85.57 ± 0.93	79.34 ± 0.77
	GRNN	93.47 ± 0.35	88.21 ± 1.35	84.77 ± 0.62	76.93 ± 0.35	89.50 ± 0.42	91.37 ± 0.27
	VGRNN	$\mathbf{95.17 \pm 0.41}$	$\mathbf{89.74 \pm 1.31}$	$\mathbf{87.32 \pm 0.60}$	$\mathbf{81.41 \pm 0.53}$	$\mathbf{91.35 \pm 0.77}$	$\mathbf{92.92 \pm 0.28}$
	SI-VGRNN	$\mathbf{96.31 \pm 0.72}$	$\mathbf{89.90 \pm 1.06}$	$\mathbf{87.69 \pm 0.92}$	$\mathbf{83.20 \pm 0.57}$	$\mathbf{91.42 \pm 0.86}$	$\mathbf{94.44 \pm 0.52}$

and edge sets. Among these four, two (**DynRNN** and **DynAERNN** [11]) are based on RNN models. By comparing our models to these methods, we will be able to see how much improvement we may obtain by improving the backbone RNN with our new prior construction compared to these RNNs with deterministic hidden states. We also compare our methods against a deep autoencoder with fully connected layers (**DynAE** [11]) to show the advantages of RNN based sequential learning methods. More detailed descriptions of these selected competing methods are described in the supplements.

Evaluation tasks. In the dynamic graph embedding literature, the term *link prediction* has been used with different definitions. While some of the previous works focused on link prediction in a transductive setting and others proposed inductive models, our models are capable of working in both settings. We evaluate our proposed models on three different *link prediction* tasks that have been widely used in the dynamic graph representation learning studies. More specifically, given partially observed snapshots of a dynamic graph $\mathcal{G} = \{G^{(1)}, \ldots, G^{(T)}\}$ with node attributes $\mathcal{X} = \{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(T)}\}$, dynamic link prediction problems are defined as follows:

- **Dynamic link detection:** Detect unobserved edges in $G^{(T)}$.

- **Dynamic link prediction:** Predict edges in $G^{(T+1)}$.

- **Dynamic new link prediction:** Predict edges in $G^{(T+1)}$ that are not in $G^{(T)}$.

Note that dynamic link detection problem can be addressed as either transductive (in-sample graphs) or inductive (out-of-sample graphs) problem while the two other link prediction problems are inherently inductive.

Experimental setups. For performance comparison, we evaluate different methods based on their ability to correctly classify true and false edges. For dynamic link detection problem, we randomly remove 5% and 10% of all edges at each time for validation and test sets, respectively. We also randomly select the equal number of non-links as validation and test sets to compute average precision (AP) and area under the ROC curve (AUC) scores. For dynamic (new) link prediction, all (new) edges are set to be true edges and the same number of non-links are randomly selected to compute AP and AUC scores. For inductive problems, in all of the datasets, we test the model on the last three snapshots of dynamic graphs while learning the parameters of the models based on the rest of the snapshots except for HEP-TH where we test the model on the last 10 sanpshots. For the datasets without node attributes, we consider the N_t-dimensional identity matrix as node attributes at time t where N_t is the number of nodes at time t.

For all datasets, we set up our VGRNN model to have a single recurrent hidden layer with 32 GRU units. All φ's in equations (3), (4), and (6) are modeled by a 32-dimensional fully-connected layer. We use two 32-dimensional fully-connected layers for φ^{prior} in (2) and 2-layer GCN with sizes equal to [32, 16] to model $\boldsymbol{\mu}_{\text{enc}}^{(t)}$ and $\boldsymbol{\sigma}_{\text{enc}}^{(t)}$ in (6). For SI-VGRNN, a stochastic GCN layer with size 32 and an additional GCN layer of size 16 are used to model the $\boldsymbol{\mu}$. The dimension of injected standard Gaussian noise ϵ is 16. The covariance matrix $\boldsymbol{\Sigma}$ is deterministic and is inferred through two layers of GCNs with sizes equal to [32, 16]. For fair comparison, the number of parameters are the same for the competing methods. In all experiments, we train the models for 1500 epochs with the learning rate 0.01. We use the validation set for the early stopping. The supplement contains additional implementation details with hyperparmaeter selection. We implemented (SI-)VGRNN in PyTorch [28].

4.1 Results and discussion

Dynamic link detection. Table 2 summarizes the results for inductive link detection in different datasets. Our proposed methods, VGRNN and SI-VGRNN, outperform competing methods across all datasets by large margins. Improvement made by (SI-)VGRNN compared to GRNN and DynAERNN supports our claim that latent random variables carry more information than deterministic hidden states specially for dynamic graphs with complex temporal changes. We note that GRNN outperforms DynAERNN due to the superior capability of GCN in capturing graph topology compared to fully connected layers. It clearly shows the advantages of the imposed latent random variables for modelling complex dynamic graphs when comparing VGRNN with GRNN.

Comparing SI-VGRNN with VGRNN shows that the Gaussian latent distribution may not always be the best choice for latent node representations. SI-VGRNN with flexible variational inference can learn more complex latent structures. The results for the Cora dataset, which also includes attributes, clearly magnify the benefits of flexible posterior as SI-VGRNN improves the accuracy by 2% compared to VGRNN. We also note that the improvement made by SI-VGRNN compared to VGRNN is marginal in Facebook dataset. The

Table 3: AUC and AP scores of dynamic link prediction on real-world dynamic graphs.

Metrics	Methods	Enron	COLAB	Facebook	Social Evo.
AUC	DynAE	74.22 ± 0.74	63.14 ± 1.30	56.06 ± 0.29	65.50 ± 1.66
	DynRNN	86.41 ± 1.36	75.7 ± 1.09	73.18 ± 0.60	71.37 ± 0.72
	DynAERNN	87.43 ± 1.19	76.06 ± 1.08	76.02 ± 0.88	73.47 ± 0.49
	VGRNN	**93.10 ± 0.57**	**85.95 ± 0.49**	**89.47 ± 0.37**	**77.54 ± 1.04**
	SI-VGRNN	**93.93 ± 1.03**	**85.45 ± 0.91**	**90.94 ± 0.37**	**77.84 ± 0.79**
AP	DynAE	76.00 ± 0.77	64.02 ± 1.08	56.04 ± 0.37	63.66 ± 2.27
	DynRNN	85.61 ± 1.46	78.95 ± 1.55	75.88 ± 0.42	69.02 ± 1.71
	DynAERNN	89.37 ± 1.17	81.84 ± 0.89	78.55 ± 0.73	71.79 ± 0.81
	VGRNN	**93.29 ± 0.69**	**87.77 ± 0.79**	**89.04 ± 0.33**	**77.03 ± 0.83**
	SI-VGRNN	**94.44 ± 0.85**	**88.36 ± 0.73**	**90.19 ± 0.27**	**77.40 ± 0.43**

Table 4: AUC and AP scores of dynamic new link prediction on real-world dynamic graphs.

| Metrics	Methods	Enron	COLAB	Facebook	Social Evo.
AUC	DynAE	66.10 ± 0.71	58.14 ± 1.16	54.62 ± 0.22	55.25 ± 1.34
	DynRNN	83.20 ± 1.01	71.71 ± 0.73	73.32 ± 0.60	65.69 ± 3.11
	DynAERNN	83.77 ± 1.65	71.99 ± 1.04	76.35 ± 0.50	66.61 ± 2.18
	VGRNN	**88.43 ± 0.75**	**77.09 ± 0.23**	**87.20 ± 0.43**	**75.00 ± 0.97**
	SI-VGRNN	**88.60 ± 0.95**	**77.95 ± 0.41**	**87.74 ± 0.53**	**76.45 ± 1.19**
AP	DynAE	66.50 ± 1.12	58.82 ± 1.06	54.57 ± 0.20	54.05 ± 1.63
	DynRNN	80.96 ± 1.37	75.34 ± 0.67	75.52 ± 0.50	63.47 ± 2.70
	DynAERNN	85.16 ± 1.04	77.68 ± 0.66	78.70 ± 0.44	65.03 ± 1.74
	VGRNN	**87.57 ± 0.57**	**79.63 ± 0.94**	**86.30 ± 0.29**	**73.48 ± 1.11**
	SI-VGRNN	**87.88 ± 0.84**	**81.26 ± 0.38**	**86.72 ± 0.54**	**73.85 ± 1.33**
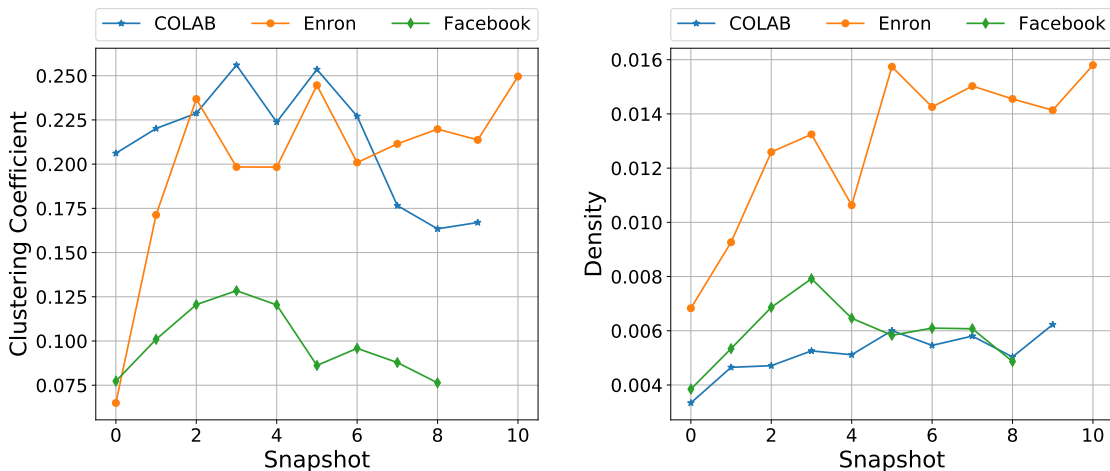
Figure 2: Evolution of graph statistics through time.

reason could be that Gaussian latent variables already represent the graph well. Therefore, more flexible posteriors do not enhance the performance significantly.

Dynamic (new) link prediction. Tables 3 and 4 show the results for link prediction and new link prediction, respectively. Since GRNN is trained as an autoencoder, it cannot predict edges in the next snapshot. However, in (SI-)VGRNN, the prior construction based on previous time steps allows us to predict links in the future. Note that none of the methods can predict new nodes, therefore, HEP-TH, Cora and Citeseer datasets are not evaluated for these tasks. VGRNN and SI-VGRNN outperform the competing methods significantly in both tasks for all of the datasets which proves that our proposed models have better generalization, which is the result of including random latent variables in our model. We note that our proposed methods improve new link prediction more substantially which shows that they can capture temporal trends better than the competing methods.

Comparing VGRNN with SI-VGRNN shows that the prediction results are almost the same for all datasets. The reason is that although the posterior is more flexible in SI-VGRNN, the prior on which our predictions are based, is still Gaussian, hence the improvement is marginal. A possible avenue for further improvements is constructing more flexible priors such as semi-implicit priors proposed by Molchanov et al. [27], which we leave for future studies.

To find out when VGRNN and SI-VGRNN show more improvements compared to the baselines, we take a closer look at three of the datasets. Figure 2 shows the temporal evolution of density and clustering coefficients of COLAB, Enron, and Facebook datasets. Enron shows the highest density and clustering coefficients, indicating that it contains dense clusters who are densely connected with each other. COLAB have low density and high clustering coefficients across time, which means that although it is very sparse but edges are mostly within the clusters. Facebook, which has both low density and clustering coefficients, is very sparse with almost no clusters. Looking back at (new) link prediction results, we see that the improvement margin of (SI-)VGRNN compared to competing methods is more substantial for Facebook. Moreover, the improvement margin diminishes when the graph has more clusters and is more dense. Predicting the evolution very sparse graphs with no clusters is indeed a very difficult task (arguably more difficult than dense graphs), in which our proposed (SI-)VGRNN is very successful. The stochastic latent variables in our models can capture the temporal trend while other methods tend to overfit very few observed links.

6

5   Conclusion

We have proposed VGRNN and SI-VGRNN, the first node embedding methods for dynamic graphs that embed each node to a random vector in the latent space. We argue that adding high level latent variables to graph recurrent neural networks not only increases its expressiveness to better model the complex dynamics of graphs, but also generates interpretable random latent representation for nodes. SI-VGRNN is also developed by combining VGRNN and semi-implicit variational inference for fexible variational inference. We have tested our proposed methods on dynamic link prediction tasks and they outperform competing methods substantially, specially for very sparse graphs.

Supplementary Material

This section contains the detailed discussion of related works, the derivation of the ELBO lower bound for SI-VGRNN inference, additional dataset details as well as experimental setups and implementation details.

A   Related Works

Several dynamic graph embedding methods have been recently developed. Instead of analyzing only node attribute dynamics on static graphs using the graph convolutional recurrent network (GCRN) [35], DyREP [39] has been proposed to capture fine-grained temporal dependencies among these evolving processes with the time stamped edges for the graphs. However, it cannot model the deletion of nodes or edges, limiting its model generalizability. Similar to DyRep [39], Know-Evolve [38] models the occurrence of edges in time using a distribution, usually a temporal point process. The latent representations of nodes are learned by deriving maximum likelihood estimates. Other methods such as DynamicTriad [45], DynGEM [10], and TIMERS [43] assume that the temporal patterns of evolving processes are of short duration and only consider the snapshot graph at the previous time step for link prediction. However, these shallow models have shortcomings in capturing long-range temporal dependencies. Furthermore, DynGEM [10] and TIMERS [43] make the assumption that the changes are smooth and use a regularization to avoid abrupt changes, again limiting model flexibility. Goyal et al. [11] have proposed using recurrent neural networks in combination with deep autoencoder to model long-term dependencies as well as sudden changes between graphs over time. Sankar et al. [33] have used structural and temporal self-attention networks to characterize the dynamics of the evolving graphs. While using recurrent neural networks (RNNs) and self-attention networks makes the transition matrix more flexible, these methods only care about the topological changes over time but do not model node attribute dynamics or complex dependencies between two evolving processes.

Existing RNN-based dynamic graph embedding methods can also be divided into two approaches: 1) Node attribute sequence approaches, where the evolution of a dynamic graph is observed as a collection of sequence node attributes. These approaches including GCRNs [35] tend to preserve very limited structural information where the adjacency matrix is not changing by time and therefore not suitable for dynamic graphs with growing/shrinking node and edge sets. 2) Adjacency matrix sequence approaches, where the evolution of a dynamic graph is observed as a collection of adjacency matrices [11]. While these approaches are capable of capturing topological changes, they do not consider node attribute dynamics. The performance of the proposed model can potentially be improved further using ideas from Bayesian multi-domain learning [14], piecewise Stationary Model [20], and sparse encoders [22].

B   Lower Bound for ELBO in SI-VGRNN

SI-VGRNN posterior can be derived by marginilizing out the mixing distribution as follows,

$$\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t), \qquad \boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)}, \mathbf{Z}^{(<t)}) = q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t | \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}),$$

$$g_{\boldsymbol{\phi}}(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}) = \int_{\boldsymbol{\psi}_t} q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t)\, q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})\, d\boldsymbol{\psi}_t\,.$$

Based on the first theorem in Yin and Zhou [42], which shows that

$$\mathbf{KL}(\mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})}[q(\mathbf{Z} \,|\, \psi)] \,||\, p(\mathbf{Z})) \leq \mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})}[\mathbf{KL}(q(\mathbf{Z} \,|\, \psi) \,||\, p(\mathbf{Z}))],$$

the lower bound for ELBO can be derived as follows,

$$\underline{\mathcal{L}} = \sum_{t=1}^{T} \underline{\mathcal{L}} \left(q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t),\, q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}) \right)$$

$$= \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbb{E}_{\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t)} \log \left(\frac{p(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)}, \mathbf{h}_{t-1})\, p(\mathbf{Z}^{(t)} \,|\, \mathbf{h}_{t-1})}{q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t)} \right)$$

$$= - \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbf{KL} \left(q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t) \,||\, p(\mathbf{Z}^{(t)} \,|\, \mathbf{h}_{t-1}) \right)$$

$$\qquad\qquad + \mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbb{E}_{\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t)} \log p(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)}, \mathbf{h}_{t-1})$$

$$\leq - \sum_{t=1}^{T} \mathbf{KL} \left(\mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t) \,||\, p(\mathbf{Z}^{(t)} \,|\, \mathbf{h}_{t-1}) \right)$$

$$\qquad\qquad + \mathbb{E}_{\boldsymbol{\psi}_t \sim q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbb{E}_{\mathbf{Z}^{(t)} \sim q(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t)} \log p(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)}, \mathbf{h}_{t-1})$$

$$= \sum_{t=1}^{T} \mathbb{E}_{\mathbf{Z}^{(t)} \sim g_{\boldsymbol{\phi}}(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \log \left(\frac{p(\mathbf{A}^{(t)} \,|\, \mathbf{Z}^{(t)}, \mathbf{h}_{t-1})\, p(\mathbf{Z}^{(t)} \,|\, \mathbf{h}_{t-1})}{g_{\boldsymbol{\phi}}(\mathbf{Z}^{(t)} \,|\, \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \right)$$

$$= \mathbb{E}_{\mathbf{Z} \sim q(\mathbf{Z}^{(\leq t)} \,|\, \mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)})} \left[\log p(\mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)}, \mathbf{Z}^{(\leq t)}) - \log q(\mathbf{Z}^{(\leq t)} \,|\, \mathbf{A}^{(\leq t)}, \mathbf{X}^{(\leq t)}) \right]$$

$$= \mathcal{L}$$

While a Monte Carlo estimation of $\underline{\mathcal{L}}$ only requires $q_{\boldsymbol{\phi}}(\mathbf{Z}^{(t)} \,|\, \boldsymbol{\psi}_t)$ to have an analytic density functions and $q_{\boldsymbol{\phi}}(\boldsymbol{\psi}_t \,|\, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})$ to be convenient to sample from, the marginal posterior $g_{\boldsymbol{\phi}}(\mathbf{Z}^{(t)} \,|\, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})$ is often

intractable and so the Monte Carlo estimation of the ELBO $\mathcal{L}$ is prohibited. SI-VGRNN evaluates the lower bound separately from the distribution sampling. This captures the idea that combining an explicit $q_\phi(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t)$ with an implicit $q_\phi(\boldsymbol{\psi}_t \mid \mathbf{X}^{(t)}, \mathbf{h}_{t-1})$ is as powerful as needed, but makes the computation tractable.

As discussed in [42], if optimizing the variational parameter by climbing $\underline{\mathcal{L}}$, without stopping the optimization algorithm early, $q_\phi(\boldsymbol{\psi}_t \mid \mathbf{X}^{(t)}, \mathbf{h}_{t-1})$ could converge to a point mass density, making SI-VGRNN degenerate to VGRNN. To prevent this problem and inspired by SIVI, we add a regularization term to the lower bound as follows,

$$\underline{\mathcal{L}}_K = \underline{\mathcal{L}} + B_K$$

where

$$B_K = \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{\psi}_t, \boldsymbol{\psi}_t^{(1)}, \ldots, \boldsymbol{\psi}_t^{(K)} \sim q_\phi(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})} \mathbf{KL}(q(\mathbf{Z}^{(t)} \mid \boldsymbol{\psi}_t) \| \tilde{g}_K(\mathbf{Z}^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})),$$

$$\tilde{g}_K(\mathbf{Z}^{(t)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})) = \frac{q_\phi(\boldsymbol{\psi}_t \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}) + \sum_{k=1}^{K} q_\phi(\boldsymbol{\psi}_t^{(k)} \mid \mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1})}{K+1}.$$

The lower bound leads to an asymptotically exact ELBO that satisfies $\underline{\mathcal{L}}_0 = \underline{\mathcal{L}}$ and $\lim_{K \to \infty} \underline{\mathcal{L}}_K = \mathcal{L}$.

C   Additional Dataset Details

Enron emails (Enron). This graph constructed from 500,000 email messages exchanged between 184 Enron employees from 1998 to 2002 [30]. The nodes represent the employees and the edges are emails exchanged between two employees. Following the same procedure as in [41, 31] we clean the data to get 10 temporal snapshots of the graph. This graph does not have any node or edge attribute.

Collaboration (COLAB). This dataset represents collaborations between 315 authors. Each node in this dynamic graph is an author and the edges represent co-authorship relationships. The data, provided by Rahman and Al Hasan [31], are collected from years 2000-2009 with a total of 10 snapshots considering each year as a time stamp. This COLAB graph does not have any node or edge attribute.

Facebook. The Facebook wall posts dynamic graph, provided by [40], has 9 time stamps. Following the same data cleaning procedure as in [41, 31], we get 663 nodes at each snapshot. No node or edge attribute is provided for this graph.

HEP-TH. The original dataset [7] covers all the citations of the papers in High Energy Physics Theory conference from January 1993 to April 2003 [25]. For each month, we create a citation graph using all the papers published up to that month. We only consider the first ten months leading to 10 snapshots in this dynamic graph. The graph has 1199 nodes at the first month and 2462 at the last one. This graph also has no node or edge attributes.

Cora. The Cora dataset is another citation graph consists of 2708 scientific publications [34]. The nodes in the graph represent the publications and the edges indicate the citation relations. Each node is provided with a 1433-dimensional binary attribute vector. Each dimension of the attribute vector indicates the presence of a word in the publication from a dictionary. Similar as the Citeseer dataset, Cora is a static graph. We similarly preprocess the data and take the indices of nodes as their arriving order. We start with the first 708 nodes and add 200 nodes and their corresponding edges to the graph at a time. The final dynamic graph includes 11 temporal snapshots, starting with 708 nodes and reaching to 2708 nodes at the last snapshot.

Social Evolution. The social evolution dataset is collected from Jan 2008 to June 30, 2009 and released by MIT Human Dynamics Lab [39]. For this dataset, we consider Calls and SMS records between users as node attributes and all Close Friendship records and Proximity as graph topology. We consider the collected information from Jan 2008 until Sep 10, 2008 (i.e. survey date) to form the initial network. We used cumulative data for 10 days periods of to form a snapshot of dynamic network for 27 snapshots.

D   Experimental Setup and Hyperparametrs Selection

Dynamic autoencoder (DynAE) [11]. This autoencoder model uses multiple fully connected layers for both encoder and decoder to capture highly non-linear interactions between nodes at each time step and across multiple time steps. It can take a set of graphs with different adjacency matrices. This model has $\mathcal{O}(nld_1)$ parameters, where n, l, and d_1 are the number of nodes, autoregressive lag, and dimension of the first hidden layer, respectively. Learning to optimize this huge number of parameters can be challenging for sparse graphs [11], which is often the case when studying real-world datasets. The input to this model at each node is the neighborhood vector of that node.

Dynamic recurrent neural network (DynRNN) [11]. This model uses LSTM networks as both encoder and decoder to capture the long-term dependencies in dynamic graphs. Comparing to DynAE, the number of parameters is reduced and the model is capable of learning complex temporal patterns more efficiently. The input to this model at each node is the neighborhood vector of that node.

Dynamic autoenncoder recurrent neural network (DynAERNN) [11]. Instead of passing the input adjacency matrices into LSTM, DynAERNN uses a fully connected encoder to initially acquire low dimensional hidden representations and then pass them as the input of LSTM to learn the embedding. The decoder of this model is a fully connected network similar to DynAE. The input to this model at each node is the neighborhood vector of that node.

Experimental setups. For VGAE at each snapshot, we use two GCN layers with 32 and 16 units for $\mathbf{GCN}_\mu$ and $\mathbf{GCN}_\sigma$. Since VGAE is a method for static graph embedding, we start training with the first snapshot and use the inferred parameters as initialization for the next snapshot. We continue this process

until the last training snapshot. In all VGAE experiments, the learning rate is set to be 0.01. We learn the model for 500 training epochs and use the validation set for the early stopping. We use the code provided by the author [23] in our experiments. For DynAE, DynRNN, and DynAERNN, we chose the dimension and number of layers of the encoder and decoder such that the total numbers of parameters is comparable to (SI-)VGRNN. For these methods, we use the source code published by the authors. In these methods, the learning rate is set to be 0.01 and the learning procedure converges in 250 training epochs. The *look back* parameter in these models, which indicates how much in the past the model looks to learn the embedding, is set to be 2. In all of the experiments in this paper, the embedding dimension is set to 16 except for HEP-TH where embedding dimension is 32.

All of the node embedding methods for link prediction performance comparison are run on a single cluster node with dual-GPU Tesla K80 accelerator and 128GB RAM. For running each epoch on the HEP-TH dataset using one of the GPUs on this cluster, SI-VGRNN, VGRNN, DynRNN, DynAERNN, and DynAE take around 36, 12, 40, 5, and 1 seconds, respectively. This is expected as DynRNN has two 2-layer LSTMs as decoders and encoders. On the other hand, the number of parameters in DynAERNN, which includes just one 2-layer LSTM, is less than that of DynRNN. DynAE are faster as they do not have LSTM units.

References

[1] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1ZdKJ-0W`.

[2] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[4] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *International ACM Conference on Knowledge Discovery and Data Mining (KDD)*, volume 24, 2018.

[5] Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2199–2207. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6039-sequential-neural-models-with-stochastic-layers.pdf`.

[6] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pages 2199–2207, 2016.

[7] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. Overview of the 2003 kdd cup. *ACM SIGKDD Explorations Newsletter*, 5(2):149–151, 2003.

[8] Alias Parth Goyal, Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6713–6723. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7248-z-forcing-training-stochastic-recurrent-networks.pdf`.

[9] Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pages 6713–6723, 2017.

[10] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[11] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 2019.

[12] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[13] Ehsan Hajiramezanali, Siamak Zamani Dadaneh, Paul de Figueiredo, Sing-Hoi Sze, Mingyuan Zhou, and Xiaoning Qian. Differential expression analysis of dynamical sequencing count data with a gamma markov chain. *arXiv preprint arXiv:1803.02527*, 2018.

[14] Ehsan Hajiramezanali, Siamak Zamani Dadaneh, Alireza Karbalayghareh, Mingyuan Zhou, and Xiaoning Qian. Bayesian multi-domain learning for cancer subtype discovery from next-generation sequencing count data. In *Advances in Neural Information Processing Systems*, pages 9115–9124, 2018.

[15] Ehsan Hajiramezanali, Mahdi Imani, Ulisses Braga-Neto, Xiaoning Qian, and Edward R Dougherty. Scalable optimal bayesian classification of single-cell trajectories under regulatory model uncertainty. *BMC genomics*, 20(6):435, 2019.

[16] Mohammadehsan Hajiramezanali and Hamidreza Amindavar. Maneuvering target tracking based on sde driven by garch volatility. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 764–767. IEEE, 2012.

[17] Mohammadehsan Hajiramezanali and Hamidreza Amindavar. Maneuvering target tracking based on combined stochastic differential equations and garch process. In *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 1293–1297. IEEE, 2012.

[18] Mohammadehsan Hajiramezanali, Seyyed Hamed Fouladi, James A Ritcey, and Hamidreza Amindavar. Stochastic differential equations for modeling of high maneuvering target tracking. *ETRI Journal*, 35(5): 849–858, 2013.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[20] Arman Hasanzadeh, Xi Liu, Nick Duffield, and Krishna R. Narayanan. Piecewise Stationary Modeling of Random Processes Over Graphs With an Application to Traffic Prediction. *arXiv preprint arXiv:1711.06954*, 2017.

[21] Arman Hasanzadeh, Ehsan Hajiramezanali, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit graph variational auto-encoders. *arXiv preprint arXiv:1908.07078*, 2019.

[22] Arman Hasanzadeh, Nagaraj T Janakiraman, Vamsi K Amalladinne, and Krishna R Narayanan. Spatially-coupled neural network architectures. *arXiv preprint arXiv:1907.02051*, 2019.

[23] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[25] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[26] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396. ACM, 2017.

[27] Dmitry Molchanov, Valery Kharitonov, Artem Sobolev, and Dmitry Vetrov. Doubly semi-implicit variational inference. *arXiv preprint arXiv:1810.02789*, 2018.

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[30] Carey E Priebe, John M Conroy, David J Marchette, and Youngser Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[31] Mahmudur Rahman and Mohammad Al Hasan. Link prediction in dynamic networks using graphlet. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 394–409. Springer, 2016.

[32] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[33] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dynamic graph representation learning via self-attention networks, 2019. URL `https://openreview.net/forum?id=HylsgnCcFQ`.

[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

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

[36] Samira Shabanian, Devansh Arpit, Adam Trischler, and Yoshua Bengio. Variational bi-lstms. *arXiv preprint arXiv:1711.05717*, 2017.

[37] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[38] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *International Conference on Machine Learning*, pages 3462–3471, 2017.

[39] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

[40] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009.

[41] Kevin S Xu and Alfred O Hero. Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552–562, 2014.

[42] Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning*, pages 5660–5669, 2018.

[43] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. Timers: Error-bounded svd restart on dynamic networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[44] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[45] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.