Neural Networks 165 (2023) 596–610

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

DyVGRNN: DYnamic mixture Variational Graph Recurrent Neural Networks

Ghazaleh Niknam[a,2], Soheila Molaei[b,2], Hadi Zare[a,*], Shirui Pan[c], Mahdi Jalili[d], Tingting Zhu[b], David Clifton[b,e]

[a] Department of Data Science and Technology, University of Tehran, Iran
[b] Department of Engineering Science, University of Oxford, United Kingdom
[c] School of Information and Communication Technology, Griffith University, Australia
[d] School of Engineering, RMIT University, Australia
[e] Oxford-Suzhou Institute of Advanced Research (OSCAR), Suzhou, China

A R T I C L E I N F O

Article history:
Received 6 September 2022
Received in revised form 29 April 2023
Accepted 25 May 2023
Available online 5 June 2023

Keywords:
Dynamic graph representation learning
Dynamic node embedding
Variational graph auto-encoder
Graph recurrent neural network
Attention mechanism

A B S T R A C T

Although graph representation learning has been studied extensively in static graph settings, dynamic graphs are less investigated in this context. This paper proposes a novel integrated variational framework called DYnamic mixture Variational Graph Recurrent Neural Networks (DyVGRNN), which consists of extra latent random variables in structural and temporal modelling. Our proposed framework comprises an integration of Variational Graph Auto-Encoder (VGAE) and Graph Recurrent Neural Network (GRNN) by exploiting a novel attention mechanism. The Gaussian Mixture Model (GMM) and the VGAE framework are combined in DyVGRNN to model the multimodal nature of data, which enhances performance. To consider the significance of time steps, our proposed method incorporates an attention-based module. The experimental results demonstrate that our method greatly outperforms state-of-the-art dynamic graph representation learning methods in terms of link prediction and clustering.[1]

1. Introduction

Many real and man-made systems can be represented as graph structures where individual entities are connected through links. Graph structures play a key role in many real-world applications. The recommendation in social networks (Liu, Shi, Pierce and Ren, 2019), traffic forecasting in transportation networks (Zhao et al., 2019), and pattern recognition in biological networks (Fout, Byrd, Shariat, & Ben-Hur, 2017) are some of these applications. Due to their complexity and high dimensions, these structures are difficult to study. To deal with this problem, representation learning approaches are used (Angles & Gutierrez, 2008). These methods aim to map high-dimensional vectors to low ones in latent space so that these latent vectors capture the structural information of the graph as well as each node's features.

Downstream machine learning tasks can then use these latent vectors as feature inputs (Bacciu, Errica, Micheli, & Podda, 2020; Hamilton, 2020). For example, COOL (Molaei, Bousejin, Zare, Jalili and Pan, 2021), and GHNN (Ju et al., 2022) employ graph representations in their classification task, Modularity-aware VGAE (Salha-Galvan, Lutzeyer, Dasoulas, Hennequin, & Vazirgiannis, 2022), and GCN-LP (Mudiyanselage, Lei, Senanayake, Zhang, & Pan, 2022) in their link prediction tasks, and SOLI (Molaei, Bousejin, Zare and Jalili, 2021) in its clustering task. Although many real-world graphs, known as dynamic graphs, evolve over time, the bulk of existing graph representation learning algorithms concentrates on static graphs, in which the set of nodes and edges does not change over time. This work aims to capture the underlying dynamics of the network.

Our proposed method, "DYnamic mixture Variational Graph Recurrent Neural Networks (DyVGRNN)", integrates a variational framework with a Graph Recurrent Neural Network (GRNN) to simultaneously capture the evolution of the dynamic graph topology and node attributes. The DyVGRNN can model the addition/removal of nodes and edges in dynamic graphs and can be applied to simple or attributed networks. While conventional variational frameworks can capture hidden and hierarchical dependencies, they are tussling with multimodal data.

Multimodality arises when in a dataset with an overall population and various subpopulations, we are unable to dedicate each subpopulation to an individual observation. Mixture models such as Gaussian Mixture Models (GMM) are an absolute solution

* Corresponding author.
E-mail address: h.zare@ut.ac.ir (H. Zare).
2 Equal Contribution.
1 The source code is available at https://github.com/GhazalehNiknam/DyVGRNN.

https://doi.org/10.1016/j.neunet.2023.05.048
0893-6080/© 2023 Elsevier Ltd. All rights reserved.

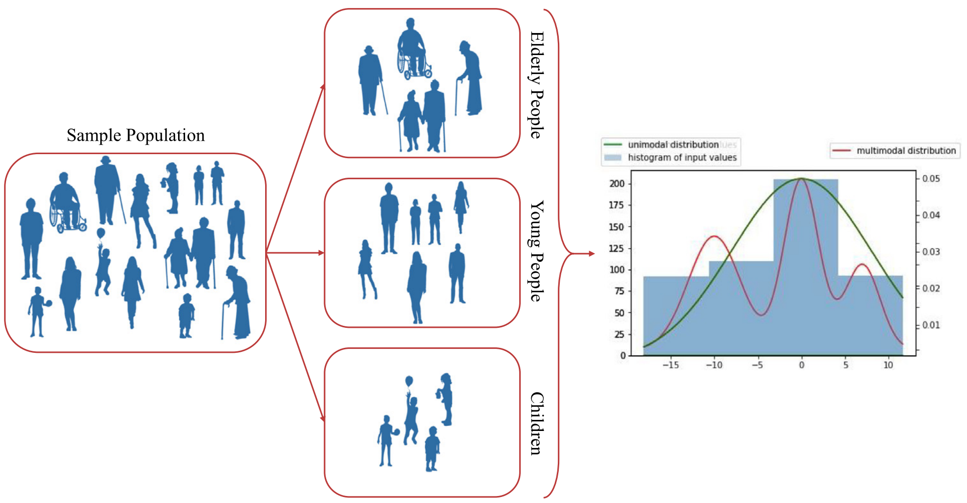


Fig. 1. Examining the effect of considering unknown subpopulations on modelling. Here, if the green curve is utilised for modelling and the age of the population under investigation is not considered, some specific information could be lost. On the other hand, a thorough knowledge of the input data is given if the red curve is employed for modelling.

for these kinds of datasets. These models describe the probability distribution of observations in the whole population (Chen & Zhang, 2020; Dilokthanakul et al., 2017; Niknam, Molaei, Zare, Clifton, & Pan, 2022). Technically, mixture models are a principled modelling approach to handle such complex data and are a universal approximator of densities (Goodfellow, Bengio, & Courville, 2016; Kostantinos, 2000).

For more clarification, consider a study that examines how an advertisement impacts a sample group of people. Some important data, like the effect of age, may be lost if the study employs the population while omitting subpopulations and models the data using a unimodal distribution. More flexibility and a more in-depth understanding of the input data can be obtained by employing a mixture model. Fig. 1 shows this affection on a synthetic dataset.

In this paper, we employ GMM to model the prior and posterior distribution in the Graph Variational Auto-Encoder (GVAE). With this combination, it is possible to capture the distribution of the input data more effectively and to get a deeper knowledge of it. Furthermore, a module based on the attention mechanism on graph snapshots is introduced in our proposed method to demonstrate the significance of time steps. Our experiments show DyVGRNN's superior performance in dynamic link prediction tasks in several real-world dynamic graphs compared to the state-of-the-art methods. Our contributions to this work are as follows:

- We propose a novel integrated variational framework consisting of extra latent random variables in structural and temporal modelling.
- We combine variational inference based on GMM with the proposed framework to infer the multimodal nature of data and improve the comprehension of the model.
- We introduce a module according to the attention mechanism of graph snapshots to consider the importance of time steps.
- Our experiments show the superior performance of the proposed DyVGRNN in several real-world dynamic graphs compared to the state-of-the-art methods.

2. Related work

To build a solid understanding of dynamic graph representation learning methods, it is important to first delve into the foundational concepts of static methods. Therefore, we will begin by exploring static methods before progressing to dynamic methods.

2.1. Static graph representation learning

Shallow embedding methods, which are based on matrix factorisation and random walks, were the first attempts to learn graph representation on static graphs. Matrix factorisation methods such as Graph Factorisation (GF) (Ahmed, Shervashidze, Narayanamurthy, Josifovski, and Smola, 2013) GraRep (Cao, Lu, & Xu, 2015), and HOPE (Ou, Cui, Pei, Zhang, & Zhu, 2016) are inspired by dimensionality reduction techniques. The key distinction among these three methods is the measure used to determine node similarity. On the other hand, in random walk methods (Grover & Leskovec, 2016; Perozzi, Al-Rfou, & Skiena, 2014), nodes have similar representations when they tend to occur together in short random walks on the graph. In contrast to matrix factorisation approaches that use deterministic node similarity measures, random walk methods use flexible stochastic node similarity measures.

DeepWalk (Perozzi et al., 2014), and node2vec (Grover & Leskovec, 2016) fall into the random walks category, which optimise embeddings to encode random walk statistics instead of decoding deterministic measures of node similarity. While shallow embedding methods have been quite popular in the last decade, they have significant drawbacks, including the inability to handle parameter sharing, difficulty with node attributes, and transductive behaviour (Hamilton, Ying, & Leskovec, 2017b). To overcome the limitations of shallow embedding methods, Graph Neural Networks (GNNs) have been proposed as powerful deep embedding approaches (Hamilton, 2020; Hamilton et al., 2017b).

GNNs are categorised into three types: those based on Graph Recurrent Neural Networks (GRNN), those based on Graph Convolutional Networks (GCN), and those based on Graph Auto Encoders (GAE) (Skarding, Gabrys, & Musial, 2021). The first structure presented in the context of GNNs is the GRNN. This structure received little attention prior to the advent of dynamic graphs. The primary assumption in the GRNN is that messages are exchanged between nodes and their neighbours until a stable equilibrium is reached.

GCNs generalise the convolutions to graph-structured data (Kipf & Welling, 2017), which plays a leading role in the construction of many other GNNs. The GCN-based approaches extract high-level node representations by stacking multiple graph convolution layers (Wu et al., 2020). Following GCNs, GAE-based methods are presented which include an encoder (mainly based on GCN) to learn representations and a decoder to reconstruct input data (Kingma & Welling, 2014; Kipf & Welling, 2017). Variational Graph Auto-Encoder (VGAE) is a variant of GAE comprising a probabilistic encoder and a probabilistic decoder to model the uncertainty of node representation for more generalisation of inference (Kipf & Welling, 2016).

2.2. Dynamic graph representation learning

Dynamic graphs can be represented in two different ways: discretely and continuously. A discrete dynamic graph is represented as a set of static graphs taken at predetermined intervals, referred to as snapshots. Continuous graphs contain no summarisation and provide whole temporal information. Continuous methods cannot be utilised on discrete networks, whereas discrete methods can be applied on continuous networks. Therefore, discrete techniques are more flexible than continuous ones (Skarding et al., 2021). While the discrete representation learning approach is our focus, we also briefly touch on the continuous representation learning approaches.

Continuous Methods. Continuous dynamic graph representation learning approaches are categorised into two groups: RNN-based and temporal point-based approaches. RNNs are used in the first category to continually maintain node embeddings. Every time an event or network change occurs, RNN-based approaches all update the embeddings of the interacting nodes. DyGNN (Ma, Guo, Ren, Tang, & Yin, 2020) falls into this category, which consists of two components: an update component that updates the states of the nodes involved in an interaction and a propagation component that propagates the update to those nodes' neighbours. JODIE (Kumar, Zhang, & Leskovec, 2019) is another RNN-based approach designed for user–item interaction networks in recommender systems. This method uses one RNN for users and the other for items. JODIE updates the embeddings when an interaction happens between a user and an item.

The utilisation of the Temporal Point Process (TPP), parametrised by neural networks, is a recurring feature of temporal point-based techniques. For example, DyREP (Skarding et al., 2021) uses a two-time scale TPP, which is parametrised by an RNN. This two-time scale TPP expresses the dynamics of the network (realised as topological evolution) as well as dynamics on the network (realised as node communication). Utilising temporal information, the attention coefficient for a structural edge between nodes is computed. Using these coefficients, the aggregate quantity required for embedding propagation is then determined. In addition, the Latent Dynamic Graph (LDG) (Kipf, Fetaya, Wang, Welling, & Zemel, 2018) extends DyREP using the Neural Relational Inference (NRI) (Han, Jiang, Wang, Ma, & Tresp, 2019) model.

Discrete Methods. The most straightforward way for modelling discrete dynamic graphs began with a single GNN in each snapshot (Skarding et al., 2021). The output of each GNN is subsequently sent into the time-series modelling module as input. For example, GCRNM1 (Seo, Defferrard, Vandergheynst, & Bresson, 2018) modelled structural features using the GCN variation described in Defferrard, Bresson, and Vandergheynst (2016) and graph evolution using the peephole LSTM introduced in Gers, Schraudolph, and Schmidhuber (2002). RgCNN (Narayan & Roe, 2018) used PATCHY-SAN, a GCN-based approach for modelling structural properties, and stacked this with a standard LSTM for modelling temporal properties.

DyGGNN (Taheri, Gimpel, & Berger-Wolf, 2019) leveraged a Gated Graph Neural Network (GGNN) and a long short-term memory network (LSTM) in its framework to model the topology of dynamic graphs and temporal information among them. Waterfall Dynamic-GCN and Concatenated Dynamic-GCN (Manessi, Rozza, & Manzo, 2020) are two architectures exploiting a GCN and an LSTM in the stacked form by applying them to each node separately. The extra skip connection of the GCN in the Concatenated Dynamic-GCN distinguishes these designs. Also, DySAT (Sankar, Wu, Gou, Zhang, & Yang, 2020) is another stacked architecture that uses self-attention blocks to capture structural and temporal properties.

The techniques mentioned earlier all offer a stacked architecture with a separate GNN for processing each snapshot of the dynamic graph and a time series module for processing the outputs of these GNNs. By integrating structural and temporal modelling into a single layer and capturing both concurrently, dynamic graphs can better capture growing relationships (Skarding et al., 2021). EvolveGCN (Pareja et al., 2020) is an integrated framework consisting of a GCN and an RNN that GCN's weights are updated with the RNN.

Another integrated framework is GC-LSTM (Chen, Wang, & Xu, 2022), which combines an LSTM with a GCN. The graph snapshots are fed into LSTM in this framework, and then a spectral graph convolution is performed on the hidden layer of LSTM. LRGCN (Li et al., 2019) leverages an R-GCN to jointly address intra-time and inter-time relationships and an LSTM to capture the time dependency between graph snapshots. Recurrent Event Network (RE-NET) (Jin, Qu, Jin, & Ren, 2020) is an auto-regressive architecture for modelling dynamic knowledge graphs and integrating an R-GCN in several RNNs.

Inspired by the success of the static GAE framework, dynamic GAE-based methods have emerged. The Dynamic Graph Embedding model (DynGEM) (Goyal, Kamra, He, & Liu, 2018) modifies the static GAE to initialise it with the weights of the previous snapshot, and substantial modifications are not permitted from one snapshot to the next. Based on DynGEM, Dyngraph2vec (Goyal, Chhetri, & Canedo, 2020) is introduced. This framework employs the l time window that defines the l most recent snapshots for encoding. Chen et al. (2019) proposed Encoder-LSTM-Decoder (E-LSTM-D), which combines an LSTM with an encoder–decoder architecture. They stacked LSTM on GAE to learn graph evolution patterns.

All the above dynamic graph representation learning techniques employ deterministic vectors to represent each node in a low-dimensional space. These deterministic representations cannot reflect the uncertainty of the node representation. Although GAE-based methods perform effectively, they disregard data distribution and may lead to overfitting and poor representations (Charte, Charte, del Jesus, & Herrera, 2020; Pan et al., 2018). The combination of the GAE framework and deep generative models has been introduced for this purpose. Deep generative models have the ability to represent complex dependencies and interactions between input and output data by considering the distribution of data (Rezende, Mohamed, & Wierstra, 2014).

GCN-GAN (Lei, Qin, Bai, Zhang, & Yang, 2019) is a generative adversarial-based method for applying GCN to examine the topological properties of each snapshot and an LSTM to characterise the evolution of the dynamic graph. This component is a generator, while a dense feed-forward network is a discriminator. SI-VGRNN (Hajiramezanali et al., 2019) is a generative approach that uses a VGAE in each snapshot. They consider a GRNN to model the temporal evolution of the graph. Our proposed framework contains an integration of VGAE and GRNN by exploiting a novel attention mechanism. Moreover, a natural assumption of multimodality of observed data is applied in our modelling (Goodfellow et al., 2016; Kostantinos, 2000).

Earlier efforts modelled the uncertainty of the observed data using a unimodal Gaussian distribution. Under this assumption,

Table 1
The notation summary. This table summarises the notations used in this paper and provides a brief explanation for each.

Symbols	Meaning
G	Dynamic graph
T	Total number of snapshots
$G^{(t)}$	A snapshot of G at time step t
$V^{(t)}$	Set of nodes in $G^{(t)}$
$E^{(t)}$	Set of edges in $G^{(t)}$
$\mathbf{A}^{(t)}$	The adjacency matrix of $G^{(t)}$
N_t	Number of nodes in $G^{(t)}$
$\mathbf{X}^{(t)}$	The features matrix of $G^{(t)}$
F	Number of features in $\mathbf{X}^{(t)}$
$\mathbf{Z}, \mathbf{W}, \mathbf{C}$	The latent variables in GMM
ϕ	The parameters of encoder neural networks
θ	The parameters of decoder neural networks
β	The parameters of the GNN related to each GMM component
ϕ_Z	The parameters of the GNN related to Z
ϕ_W	The parameters of the GNN related to W
H	The dimension of the representation embedding size

modelling complex data with properties like multimodality is inefficient. Although SI-VGRNN develops semi-implicit variational inference for greater modelling flexibility, they only regard this assumption on their posterior modelling and not their prior. Hence, the improvement in their results is marginal (Hajiramezanali et al., 2019). To capture multimodality in the input data, our proposed DyVGRNN leverages GMM to model prior and posterior.

Furthermore, most previous works treat timed snapshots equally, despite the fact that assessing differences in snapshot significance may lead to more accurate results. SI-VGRNN assigns a fixed priority to different time series modelling snapshots, even though these snapshots may affect them differently. Here, we propose an attention-based module for examining the importance of snapshots. Unlike the traditional application of the attention mechanism in static graph representation learning, where the input is a matrix of nodes and the attention mechanism examines the importance of each node's neighbouring nodes, the input in our module is a matrix of information for each time step, and the importance of time steps is examined.

3. The proposed model

3.1. Notation and problem definition

Let us represent a dynamic graph G as $G = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, where $G^{(t)} = (V^{(t)}, E^{(t)})$ denotes a graph at time step t. Here $V^{(t)}$ and $E^{(t)}$ represent sets of nodes and edges, and T denotes the number of time steps. Since we intend to model a possible node or edge set change, the number of nodes and/or edges can change over time. Thus, $(V^{(t)}, E^{(t)})$ and $(V^{(t+1)}, E^{(t+1)})$ can be completely different. The input of the proposed method is a sequence of variable-length adjacency matrices in the form of $\mathbf{A} = \{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(T)}\}$ where $\mathbf{A}^{(t)} \in R^{N_t \times N_t}$ and N_t denotes the number of nodes in this snapshot. Furthermore, there is a sequence of variable-length feature matrices in the form of $\mathbf{X} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(T)}\}$ as input, if the nodes have features. Here, each $\mathbf{X}^{(t)}$ is a $N_t \times F$ matrix, where F denotes the number of features. We assume F is constant over time. Table 1 summarises the notations used in this paper.

3.2. DyVGRNN

Fig. 2 shows a high-level overview of our proposed method, DyVGRNN. The proposed method consists of three main modules described in this section. First, integrating GMM and VGAE used to model each graph snapshot is examined. Following, the process of modelling the evolution is described. Finally, we discuss the attention-based module for considering the importance of each graph snapshot in modelling evolution over time.

3.2.1. Integration of GMM and VGAE

Our model defines three hidden variables $\mathbf{Z}$, $\mathbf{W}$, and $\mathbf{C}$ for integrating GMM and VGAE into a framework called Gaussian Mixture Variational Graph Auto Encoder (GM-VGAE). In this case, the inference model of standard VGAE for snapshot t, generalises and follows the process shown in Eq. (1)

$$\begin{aligned}
&\mathbf{W}^{(t)} \sim \mathcal{N}(0, \mathbf{I}) \\
&\mathbf{C}^{(t)} \sim Cat(\pi) \\
&\mathbf{Z}^{(t)}|\mathbf{C}^{(t)}, \mathbf{W}^{(t)} \sim \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_{c_k^{(t)}}(\mathbf{W}^{(t)}; \beta), \boldsymbol{\Sigma}_{c_k^{(t)}}(\mathbf{W}^{(t)}; \beta))^{c_k^{(t)}}
\end{aligned} \tag{1}$$

Here, K is a hyperparameter of the model, which denotes the number of components in the mixture model. $\mathbf{W}^{(t)}$ is one of the latent variables of snapshot t that follows a Gaussian distribution with mean zero and covariance matrix $\mathbf{I}$. $\mathbf{C}^{(t)}$ is a one-hot vector denoting the mixing coefficients of the Gaussian mixture components of snapshot t. This vector is sampled from π (the mixing probability), which indicates one of the Gaussian mixture components.

$\mathbf{W}^{(t)}$ is fed to a GNN parametrised by β. The output of this neural network is a set of K ($\boldsymbol{\mu}_{c_k}^{(t)}$) and K ($\boldsymbol{\Sigma}_{c_k}^{(t)}$). Each $\boldsymbol{\mu}_{c_k}^{(t)}$ and $\boldsymbol{\Sigma}_{c_k}^{(t)}$ in these sets are calculated by a GNN. An inner product between latent variables is used for reconstructing the adjacency matrix, as shown in Eq. (2).

$$\begin{aligned}
p(\mathbf{A}^{(t)}|\mathbf{Z}^{(t)}) &= \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij}^{(t)}|\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}) \\
p(A_{ij}^{(t)}|\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}) &= Sigmoid(\mathbf{z}_i^{(t)T}\mathbf{z}_j^{(t)})
\end{aligned} \tag{2}$$

Based on the mean-field variational family, the general form of posterior can be factorised as Eq. (3).

$$\begin{aligned}
&q(\mathbf{Z}^{(t)}, \mathbf{W}^{(t)}, \mathbf{C}^{(t)}|\mathbf{A}^{(t)}) = \\
&\quad \prod_{i=1}^{N_t} q_{\phi_Z}(\mathbf{z}_i^{(t)}|\mathbf{A}_i^{(t)}) q_{\phi_W}(\mathbf{w}_i^{(t)}|\mathbf{A}_i^{(t)}) q_{\beta}(\mathbf{z}_i^{(t)}|\mathbf{c}_i^{(t)}, \mathbf{w}_i^{(t)})
\end{aligned} \tag{3}$$

In this equation, ϕ_Z, ϕ_W, and β are the parameters of neural networks, and the output of these networks is the parameters of the variational distributions. The $\mathbf{C}$-posterior is as follows,

$$\begin{aligned}
p_\beta(\mathbf{c}_j = 1|\mathbf{Z}, \mathbf{W}) &= \frac{p(\mathbf{c}_j = 1)p(\mathbf{Z}|\mathbf{c}_j = 1, \mathbf{W})}{\sum_{k=1}^{K} p(\mathbf{c}_k = 1)p(\mathbf{Z}|\mathbf{c}_j = 1, \mathbf{W})} \\
&= \frac{\pi_j \mathcal{N}(\mathbf{Z}|\mu_j(\mathbf{W}; \beta), \sigma_j(\mathbf{W}; \beta))}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{Z}|\mu_k(\mathbf{W}; \beta), \sigma_k(\mathbf{W}; \beta))}
\end{aligned} \tag{4}$$

3.2.2. Modelling the evolution

In contrast to standard VGAE that samples prior from a standard Gaussian distribution ($\mathcal{N}(0, I)$), the proposed VGAE (GM-VGAE) has a new prior extraction process that allows the parameter of the prior distribution to be modelled by a function of the previous time step. In other words, the prior distribution parameters are based on the information of the previous hidden state rather than deterministic parameters. The construction of

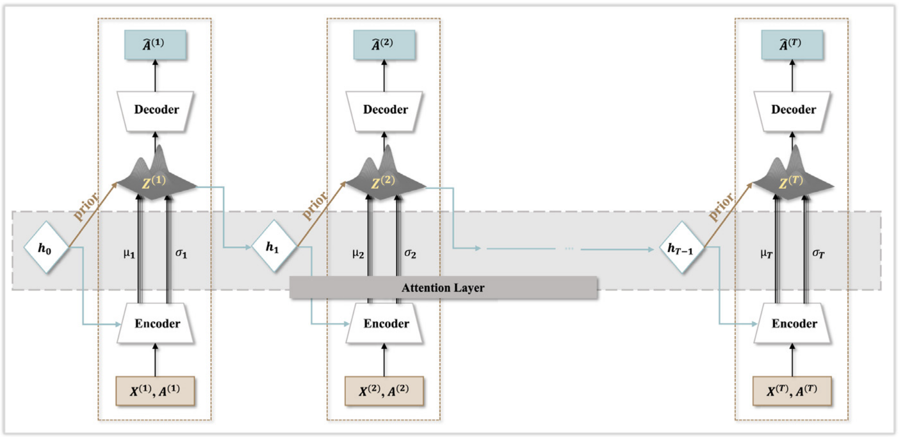


Fig. 2. A high-level overview of our method. A VGAE integrated with GMM performs on each time step. The prior distribution of the VGAE is a function of the previous time step and a GRNN structure with extra hidden variables of the prior time step acts as a backbone of the entire framework. GRNN captures the dynamics of both graph topology and the node features jointly. The hidden state of GRNN is also added to latent random variables of GM-VGAE, making it capable of modelling variations in the topology or graph properties over time. Moreover, an attention-based module measures the importance of each graph snapshot in modelling evolution over time.

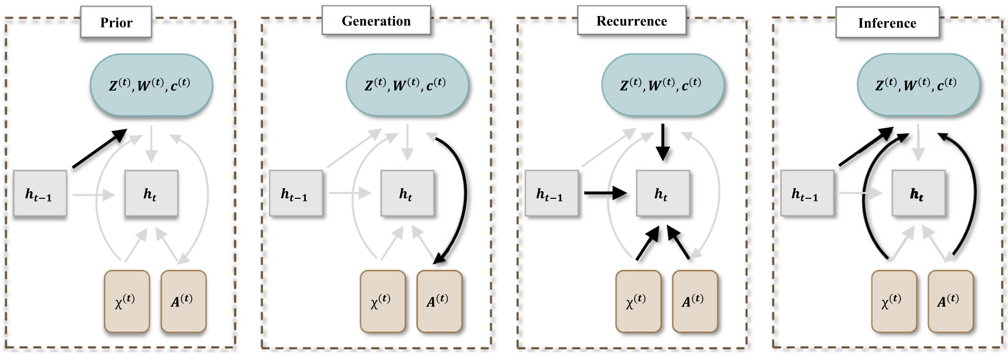


Fig. 3. Graphical illustrations for Prior, Inference, Recurrence, and Generation of DyVGRNN. Arrows indicate the dependency of each component on the other component. The drawn arrow for Prior suggests the source of prior parameters, which is the previous hidden state of the model. The arrows of Inference and Recurrence indicate the resources needed to infer the latent variables and update the hidden state, respectively. The arrow of generation shows that the adjacency matrix can be reconstructed by having latent variables.

the prior distribution can be written as shown in Eq. (5).

$$\begin{aligned}
&\{\boldsymbol{\mu}^{(t)}_{prior}, \boldsymbol{\Sigma}^{(t)}_{prior}\} = F^{prior}(\mathbf{h}_{t-1})\\
&\mathbf{W}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}^{(t)}_{prior}, \boldsymbol{\Sigma}^{(t)}_{prior})\\
&\mathbf{C}^{(t)} \sim Cat(\pi)\\
&\mathbf{Z}^{(t)}|\mathbf{C}^{(t)}, \mathbf{W}^{(t)} \sim \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_{\mathbf{c}^{(t)}_k}(\mathbf{W}^{(t)}; \beta), \boldsymbol{\Sigma}_{\mathbf{c}^{(t)}_k}(\mathbf{W}^{(t)}; \beta))^{\mathbf{c}^{(t)}_k}
\end{aligned} \tag{5}$$

Here $\boldsymbol{\mu}^{(t)}_{prior}$ and $\boldsymbol{\Sigma}^{(t)}_{prior}$ represent the parameters of the prior distribution. F^{prior} is a function that produces the parameters of prior distribution based on the previous hidden state. This function can be a neural network. The prior distribution of the first step is assumed to be a standard multivariate Gaussian distribution as $\mathcal{N}(0, \mathbf{I})$. If node addition occurs at each snapshot, the prior distribution of the added node is defined as $\mathcal{N}(0, \mathbf{I})$. Eliminating a node can be conceived as removing all edges connected to the node. In this way, prior probabilities are unaffected.

The GRNN structure acts as a chain in the whole framework to capture the dynamics of graph topology and features of the nodes. The GRNN update rule is defined as shown in Eq. (6).

$$\mathbf{h}_t = f(\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{Z}^{(t)}, \mathbf{h}_{t-1}) \tag{6}$$

Here f can be one of the Recurrent Neural Network (RNN) frameworks, such as long short-term memory (LSTM) or gated recurrent units (GRU). In this paper, we use LSTM-Attention for this purpose. If node addition occurs at snapshot t, the hidden state of the node at snapshot $t-1$ is considered being zero. The Z-posterior of the model is shown in Eq. (7).

$$\begin{aligned}
&q(\mathbf{Z}^{(t)}|\mathbf{A}^{(t)}, \mathbf{X}^{(t)}, \mathbf{h}_{t-1}) \sim \prod_{k=1}^{K} N(\mu_{c^{(t)}_{k,enc}}, \Sigma_{c^{(t)}_{k,enc}})^{\mathbf{c}^{(t)}_k}\\
&\boldsymbol{\mu}^{(t)}_{enc} = GNN_{\mu}(A^{(t)}, CONCAT(\mathbf{X}^{(t)}, \mathbf{h}_{t-1}))\\
&\boldsymbol{\Sigma}^{(t)}_{enc} = GNN_{\Sigma}(A^{(t)}, CONCAT(\mathbf{X}^{(t)}, \mathbf{h}_{t-1}))
\end{aligned} \tag{7}$$

Here $\mu^{(t)}_{enc}$ and $\Sigma^{(t)}_{enc}$ represent the parameters of the posterior distribution, respectively. $GNN_{\mu}(.)$ and $GNN_{\Sigma}(.)$ can be any kind of GNN. We use a two-layer GCN for this purpose. The graphical illustrations for Prior, Inference, Recurrence, and Generation of DyVGRNN are shown in Fig. 3. To carry out the learning process, the standard ELBO formulation is generalised as Eq. (8) (Dilokthanakul et al., 2017).

$$L_{ELBO} = \mathbb{E}_q\left[\frac{p(\mathbf{A}^{(t)}, \mathbf{Z}^{(t)}, \mathbf{W}^{(t)}, \mathbf{C}^{(t)})}{q(\mathbf{Z}^{(t)}, \mathbf{W}^{(t)}, \mathbf{C}^{(t)}|\mathbf{A}^{(t)})}\right] \tag{8}$$

in which,

$$p(\mathbf{A}^{(t)}, \mathbf{Z}^{(t)}, \mathbf{W}^{(t)}, \mathbf{C}^{(t)}) = p(\mathbf{W}^{(t)})p(\mathbf{C}^{(t)})p(\mathbf{Z}^{(t)}|\mathbf{W}^{(t)}, \mathbf{C}^{(t)})p(\mathbf{A}^{(t)}|\mathbf{Z}^{(t)}) \tag{9}$$

Based on the mean-field variational family, shown in Eqs. (3) and (9), the lower bound for each snapshot can be written as Eq. (10).

$$\begin{gathered} L_{ELBO}^{(t)} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})}[\log p(\mathbf{A}^{(t)}|\mathbf{Z}^{(t)})]- \\ \mathbb{E}_{q(\mathbf{W}|\mathbf{A},\mathbf{X})p(\mathbf{C}|\mathbf{Z},\mathbf{W})}[D_{KL}(q_{\phi_Z}(\mathbf{Z}^{(t)}|\mathbf{A}^{(t)}, \mathbf{X}^{(t)})\|p_\beta(\mathbf{Z}^{(t)}|\mathbf{W}^{(t)}, \mathbf{C}^{(t)}))]- \\ D_{KL}(q_{\phi_W}(\mathbf{W}^{(t)}|\mathbf{A}^{(t)}, \mathbf{X}^{(t)})\|p(\mathbf{W}^{(t)}))- \\ \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})q(\mathbf{W}|\mathbf{A},\mathbf{X})}[D_{KL}(p_\beta(\mathbf{C}^{(t)}|\mathbf{Z}^{(t)}, \mathbf{W}^{(t)})\|p(\mathbf{C}^{(t)}))] \end{gathered} \tag{10}$$

This equation consists of four terms representing the reconstruction error term, prior conditional term, **W**-prior term, and **C**-prior term. The total loss function of the model is calculated as the sum of the loss functions of each snapshot. Thus, the loss function can be written as Eq. (11).

$$L_{ELBO}^{(total)} = \sum_{t=1}^{T} L_{ELBO}^{(t)} \tag{11}$$

3.2.3. Attention module

The attention mechanism was first introduced by Bahdanau, Cho, and Bengio (2015) in the field of Natural Language Processing (NLP). This work became the basis for Vaswani et al. (2017), which attracted much attention. Recent studies in NLP have emphasised that the use of the attention mechanism improves the efficiency and performance of models (Shen et al., 2018; Tan, Wang, Xie, Chen, & Shi, 2018; Tenenbaum, De Silva, & Langford, 2000). Other fields have also been positively influenced by the capability of this mechanism (Sankar et al., 2020; Veličković et al., 2018), and we endeavour to use the potential of this mechanism.

We add an attention module to the proposed model, which receives as input the hidden states and structural information of all time steps. The attention module's output hidden state is considered the model's final hidden state. Structural information is then used to calculate the loss function like Eq. (10) with the parameters gained by the attention mechanism. Then, backpropagation of the gradients of the loss function leads to updating the weights. In this way, the importance of each snapshot is taken into consideration in the learning process.

Here, the mean and standard deviation matrices are remarked as the structural information of each snapshot. Thereupon, the received information is converted into a matrix, each row showing one snapshot's information. This operation is fulfilled for both the mean and standard deviation matrices. The un-normalised attention scores between two snapshots are calculated according to Eq. (12).

$$\begin{aligned} e_{i,j}^{\boldsymbol{\mu}} &= LeakyReLU(a(CONCAT(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j))) \\ e_{i,j}^{\boldsymbol{\sigma}} &= LeakyReLU(a(CONCAT(\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_j))) \\ e_{i,j}^{\mathbf{h}} &= LeakyReLU(a(CONCAT(\mathbf{h}_i, \mathbf{h}_j))) \end{aligned} \tag{12}$$

Here a is a learnable weight vector. The normalised attention scores calculate by applying a Softmax to un-normalised attention scores as shown in Eq. (13). Eventually, these α sets determine the importance of each time step.

$$\begin{aligned} \alpha_{i,j}^{\boldsymbol{\mu}} &= \frac{\exp e_{i,j}^{\boldsymbol{\mu}}}{\sum_{k\in\boldsymbol{\mu}} \exp e_{i,k}^{\boldsymbol{\mu}}}, \quad \boldsymbol{\mu} = \{\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(T)}\} \\ \alpha_{i,j}^{\boldsymbol{\sigma}} &= \frac{\exp e_{i,j}^{\boldsymbol{\sigma}}}{\sum_{k\in\boldsymbol{\sigma}} \exp e_{i,k}^{\boldsymbol{\sigma}}}, \quad \boldsymbol{\sigma} = \{\boldsymbol{\sigma}^{(1)}, \boldsymbol{\sigma}^{(2)}, \ldots, \boldsymbol{\sigma}^{(T)}\} \\ \alpha_{i,j}^{\mathbf{h}} &= \frac{\exp e_{i,j}^{\mathbf{h}}}{\sum_{k\in\mathbf{h}} \exp e_{i,k}^{\mathbf{h}}}, \quad \mathbf{h} = \{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(T)}\} \end{aligned} \tag{13}$$

Table 2
Summary of the employed datasets. "–" in "Number of Edge" column means the number changes across different snapshots.

Dataset	Number of snapshots	Number of nodes	Number of edges	Number of node attributes
Enron	11	184	115–266	–
Colab	10	315	165–308	–
Facebook	6	663	844–1068	–
UCI	7	537–1899	59 835	–
Cora	6	500–2708	406–5429	1433
LFB	36	45 435	180 011	–
AS733	30	6628	13 512	–

Both DyREP (Skarding et al., 2021) and DySAT (Sankar et al., 2020) leverage the attention mechanism as part of their method. They employ node-based attention mechanisms in their framework. DyREP computes the attention coefficient and evaluates the importance of each node's neighbours using temporal information. DySAT applies one attention layer to focus on each node's immediate neighbours, and a second attention layer to focus on each node's temporal history in each snapshot. While our attention module is based on graphs, these two methods use a node-based attention module. In fact, in their methods, the input would be a matrix of nodes and the attention mechanism examines the importance of the neighbouring nodes of each node. Whereas the input of our module is a matrix of information for each time step, and the importance of time steps is examined.

4. Experimental details

In this section, the results of the experiments are presented. First, the datasets, the state-of-the-art methods, and the studied tasks and metrics are introduced. Then, the results of the experiments are described.

4.1. Datasets

Our experiments are performed on five real-world graph datasets. Table 2 presents a summary of the employed datasets.

Facebook. This dataset contains information about Facebook posts. The Facebook dataset is collected by Viswanath, Mislove, Cha, and Gummadi (2009), and the procedure of cleaning and preparing the data is similar to the procedure in Rahman and Al Hasan (2016) and Xu and Hero (2014). This dataset has 663 nodes and 1068 edges but does not contain node or edge attributes.

LFB. This dataset is a larger-scale version of the Facebook dataset containing 45 435 nodes and 180 011 edges. The procedure of cleaning and preparing the data in this version is also similar to the procedure in Rahman and Al Hasan (2016) and Xu and Hero (2014). 36 snapshots of the activations throughout the last three years are included in the dataset. In the LFB dataset, there are a large number of users but not many links between them.

Enron emails (Enron). This dataset contains 500,000 emails exchanged between Enron employees from 1998 to 2002 (Priebe, Conroy, Marchette, & Park, 2005). The nodes represent 184 employees, and the edges represent the emails exchanged between pairs of employees in the graph created from this dataset. The steps of cleaning and producing the appropriate structure for applying the algorithm are done according to the procedure in Hajiramezanali et al. (2019), Rahman and Al Hasan (2016) and Xu and Hero (2014). This dataset has no node or edge attributes.

Collaboration (Colab). There is information about co-authorship relationships between 315 authors in this dataset. Each node represents an author, and each edge demonstrates co-authorship relationships between a pair of authors from 2000 to 2009 (Rahman & Al Hasan, 2016). This dataset has no node or edge attributes.

UCI. This dataset was aggregated by the University of California, Irvine (Priebe et al., 2005). In this dataset, message interaction information between students based on an online community has been collected. Nodes represent students, and edges represent the sending of a message between two students. This information was collected over a 7-day period. Each day denotes one snapshot of the graph. This dynamic graph starts at 537 nodes, ends at 1899 nodes, has 59 835 edges, and has no node properties.

Cora. This dataset is a static citation graph in which the nodes represent the publications, and the edges denote the citation (Sen et al., 2008). Cora consists of 2708 nodes with a 1433-dimensional binary attribute vector. To make use of Cora dynamically, we preprocess the data in the same way as described in Hajiramezanali et al. (2019) and Liu et al. (2019). In the dynamic network, we added 500 nodes with their accompanying edges at each temporal snapshot (208 nodes for the last snapshot), using the indexes of the nodes as their arrival order, and six snapshots of the dynamic graph were taken, starting with 500 nodes and ending with 2708 nodes.

AS733. This dataset is a communication network containing Autonomous Systems (AS) and traffic flows between them that show who communicates with whom. AS733 was gathered from the Route Views Project at the University of Oregon, which contains 733 daily instances spanning 785 days between 1997 and 2000 (Leskovec, Kleinberg, & Faloutsos, 2005). There are 6628 nodes and 13 512 edges in this dataset.

4.2. Baselines

We compare DyVGRNN with the following baselines and state-of-the-art methods. We use the original implementation of the methods introduced in their paper. To ensure a fair comparison, the hyperparameters are adjusted based on the suggestion in their papers.

4.3. Discrete dynamic graph representation learning methods

DynAE (Dynamic Auto-Encoder) (Goyal et al., 2020): This model is an auto-encoder composed of multiple fully connected layers as the encoder and decoder. These layers are used to capture non-linear interactions between nodes at each snapshot and across multiple snapshots.

DynRNN (Dynamic Recurrent Neural Network) (Goyal et al., 2020): This model consists of an LSTM encoder and an LSTM decoder. These encoder and decoder allow capturing the long-term dependencies in dynamic graphs.

DynAERNN (Dynamic Auto-Encoder Recurrent Neural Network) (Goyal et al., 2020): This model includes a fully connected layer connected to an LSTM as the encoder. The fully connected layer generates initial low-dimensional hidden representations, which are then fed to LSTM. Here, the decoder is a fully connected network.

SI-VGRNN (Variational Graph Recurrent Neural Networks) (Hajiramezanali et al., 2019): This method was the inspiration for this paper that is based on VGAE, which is combined with GRNN to capture topology and node feature changes in dynamic graphs. This paper suggested regarding and disregarding the semi-implicit part as an SI-VGRNN and VGRNN, respectively.

DySAT (Dynamic Self-Attention Network) (Sankar et al., 2020): This method computes node representations through self-attention blocks that capture structural and temporal properties.

HTGN (Hyperbolic Temporal Graph Network) (Yang, Zhou, Kalander, Huang, & King, 2021): This approach maps the dynamic graph in hyperbolic space and combines a hyperbolic GNN and a hyperbolic GRNN to capture network evolution while implicitly maintaining hierarchical information.

4.4. Continuous dynamic graph representation learning methods

DyREP (Skarding et al., 2021): This model uses a two-time scale Temporal Point Process (TPP) model, which is parametrised by an RNN.

JODIE (Kumar et al., 2019): This model uses RNNs to predict representations in the future. Since the method was originally proposed for bipartite graphs, we modified it for standard graphs in accordance with Wang, Chang, Liu, Leskovec, and Li (2020).

TGAT (Xu, Ruan, Korpeoglu, Kumar, & Achan, 2020): This model is based on the self-attention mechanism and develops a functional time encoding technique based on the classical Bochner's theorem.

4.5. Tasks

We perform the link prediction and clustering tasks in this study to evaluate our method. The link prediction task in dynamic graphs is defined differently than in static graphs. Given a dynamic graph $G = \{G^{(1)}, G^{(2)}, \dots, G^{(T)}\}$, the link prediction is divided into two categories: (1) dynamic link prediction attempts to identify the unobserved links in $G^{(T)}$, and (2) dynamic new link prediction tries to predict links in $G^{(T+1)}$ which does not exist in $G^{(T)}$.

4.6. Metrics

We use the Average Precision (AP) and the Area Under the receiver operating characteristic Curve (AUC) (Kipf & Welling, 2016) metrics to compare our proposed method with state-of-the-art methods in link prediction and new link prediction tasks. To calculate these measures, all edges of G^T are considered as actual links (positive samples), and on the other hand, the pairs of nodes without an edge imply false links (negative samples). Furthermore, the silhouette criterion is applied for the evaluation of the clustering results to interpret and validate data consistency within clusters.

4.7. Settings

The proposed model uses the LSTM-attention with a single hidden layer of 32 units for the GRNN. The GNN_μ and GNN_Σ are set to be two-layer GCN with 32 and 16 units, respectively. Our model is initialised using Glorot initialisation (Glorot & Bengio, 2010). The learning rate for training our model is set to be 0.01. Model training is done in 1000 epochs using the Adam SGD optimiser (Kingma & Ba, 2015). Moreover, we use a validation set for the early stopping. Therefore, the training will terminate if the validation accuracy does not improve in 10 consecutive stages. The mean of the evaluation metrics is reported based on 10 runs of the model under different random seeds.

Table 3
AP scores of link prediction on dynamic graphs. The best results are highlighted.

Model	Enron	Colab	Facebook	LFB	UCI	Cora	AS733
DynAE	76.00	64.02	56.04	58.90	91.12	57.11	74.23
DynRNN	85.61	78.95	75.88	75.28	89.21	80.75	87.53
DynAERNN	89.37	81.84	78.55	78.27	89.92	82.93	88.77
DySAT	93.06	90.40	80.39	80.39	85.01	87.73	96 .72
HTGN	94.31	91.91	83.80	83.80	86.72	90.12	98.41
VGRNN	93.29	87.77	89.04	81.40	91.83	93.32	96.69
SI-VGRNN	94.44	88.36	90.19	82.01	93.16	96.68	97.13
DyVGRNN	**97.28**	**96.77**	**92.70**	**86.22**	**95.07**	**97.48**	**99.10**

Table 4
AUC scores of link prediction on dynamic graphs. The best results are highlighted.

Model	Enron	Colab	Facebook	LFB	UCI	Cora	AS733
DynAE	74.22	63.14	56.06	57.18	91.89	57.13	73.84
DynRNN	86.41	75.7	73.18	73.98	89.27	80.10	86.11
DynAERNN	87.43	76.06	76.02	75.28	90.08	78.00	88.37
DySAT	93.06	87.25	76.88	76.88	86.73	85.3	95.06
HTGN	94.17	89.26	83.70	83.7	87.25	89.73	98.75
VGRNN	93.10	85.95	89.47	79.11	92.01	94.41	95.17
SI-VGRNN	93.93	85.45	90.94	80.27	93.5	97.17	96.37
DyVGRNN	**96.59**	**95.80**	**93.17**	**86.73**	**95.15**	**98.74**	99.19

Table 5
AUC scores of new link prediction on dynamic graphs. The best results are highlighted.

Model	Enron	Colab	Facebook	LFB	UCI	Cora	AS733
DynAE	66.10	58.14	54.62	56.34	89.94	56.27	68.93
DynRNN	83.20	71.71	73.32	74.15	87.27	79.94	74.72
DynAERNN	83.77	71.99	76.35	76.55	88.29	77.36	76.63
DySAT	87.94	79.74	74.97	74.97	84.2	86.11	82.84
HTGN	91.26	81.74	82.21	82.21	84.98	87.85	96.62
VGRNN	88.43	77.09	87.20	76.33	89.93	94.94	81.86
SI-VGRNN	88.60	77.95	87.74	77.42	90.45	96.36	83.27
DyVGRNN	**94.26**	**92.71**	**92.51**	**85.26**	**94.17**	**97.16**	**97.89**

Table 6
AP scores of new link prediction on dynamic graphs. The best results are highlighted.

Model	Enron	Colab	Facebook	LFB	UCI	Cora	AS733
DynAE	66.50	58.82	54.57	54.91	89.65	56.65	69.12
DynRNN	80.96	75.34	75.52	76.01	86.86	80.01	75.12
DynAERNN	85.16	77.68	78.70	78.27	88.15	82.34	76.87
DySAT	86.83	83.47	78.34	78.34	83.94	87.15	89.07
HTGN	90.62	84.06	81.70	81.7	84.26	89.83	95.52
VGRNN	87.57	79.63	86.30	79.61	89.48	93.21	88.59
SI-VGRNN	87.88	81.26	86.72	80.12	90.07	95.32	89.49
DyVGRNN	**94.44**	**93.65**	**91.81**	**85.00**	**94.11**	**96.82**	96.83

4.8. Results analysis

Dynamic Link Prediction. Tables 3 and 4 represent the comparison results in terms of AP and AUC on the link prediction task. The results of the dominant algorithm are highlighted. DyVGRNN shows significant improvement in results compared to the other methods. The enhancement of our method using the AP criterion compared to the first method is 21.28% in the Enron dataset, 32.75% in the Colab dataset, 36.66% in the Facebook dataset, and 40.37% in the Cora dataset. Large datasets like LFB and AS733 show improvements of 27.32% and 24.87%, respectively. Likewise, in the UCI dataset, where the first method performed well, our proposed method boosts the result by 3.95%. If we compare the AUC criteria, the results are also significantly improved. For example, the results of a comparison with SI-VGRNN, which on average provided the best results among the previous methods, show that the proposed method leads to 2.66% improvement in the Enron dataset, 10.35% in the Colab dataset, 2.23% in the Facebook dataset, 1.65% in the UCI dataset, and eventually 1.57% in the Cora dataset. Large datasets LFB and AS733 have improvements of 6.46.21% and 2.82%, respectively.

Dynamic New Link Prediction. Tables 5 and 6 represent the results of comparisons regarding AUC and AP on the new link prediction task. The proposed method has achieved significant results in all datasets. A similar analysis for the link prediction task can be provided for the new link prediction task. In general, it can be noted that the proposed method can have a high potential for predicting the overall structure of the graph in the new snapshot.

To point out some significant improvements, we can mention the progress of more than 40% in the Cora dataset or the increase of over 37% in the Facebook dataset in both criteria compared to DynAE. In addition, our method performed superior to SI-VGRNN, which indicates a positive effect of the assumption of GMM and the proposed attention module. A comparison of the proposed DyVGRNN and VGRNN is presented in Appendix A in order to further analyse the effectiveness of the methods.

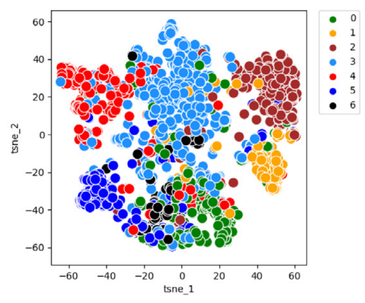
Clustering. For further investigation, we provide a clustering comparison as well. The proposed approach is compared against SI-VGRNN, which achieves the highest result among various methods, and DySAT, which performs best among deterministic ones. To this end, the silhouette criterion is utilised for clustering the Cora dataset. This criterion is 0.32 for DySAT, 0.36 for SI-VGRNN, and 0.43 for our approach. Demonstrating a transparent view, we visualise the representations of these three methods in a two-dimensional space as shown in Fig. 4. Compared to the raw features, the trained representations in two-dimensional space for our method indicate well-separated clustering compared to SI-VGRNN. In addition, modelling uncertainty in SI-VGRNN and DyVGRNN yields superior clustering outcomes compared to DySAT, which is a deterministic-based method. We also provide a classification comparison in Appendix C.

Comparison with Continuous Methods. We compare our model to state-of-the-art methods in the category of continuous dynamic graph representation learning in terms of dynamic link prediction. The results of this comparison are shown in Fig. 5. As demonstrated in Fig. 5, our proposed method outperforms other continuous methods. DyRep has the best performance among existing continuous approaches, which our method enhances.

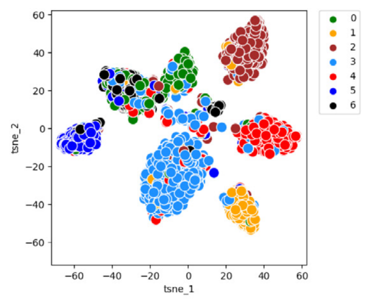
4.9. Complexity and running time

To compute the time complexity of our method, the analysis of Gao and Ribeiro (2022) is followed. For this purpose, the proposed DyVGRNN can be divided into three main parts. (1) modelling node temporal attributes by LSTM which the time complexity is $O(T|V|H^2)$. (2) modelling node structural properties by VGAE, which consists of GCN structure in its encoder and an inner product decoder. Time complexity of GCN is $O(|V|H^2+(|V|+|E|)H)$. Since H and $|V|$ are relatively small w.r.t. to $|E|$, the time cost is indeed $O(|E|)$.

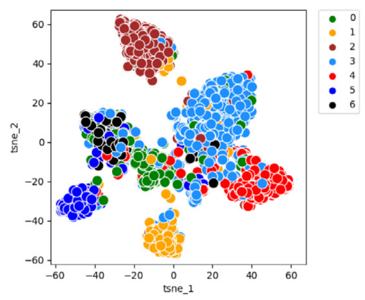
Moreover, the time complexity of the inner product decoder is $O(|E|)$. As a result, the time complexity of VGAE is $O(|E|)$. (3) Considering the attention mechanism which has an order of $O(EH^2)$. Eventually, the time complexity of our proposed



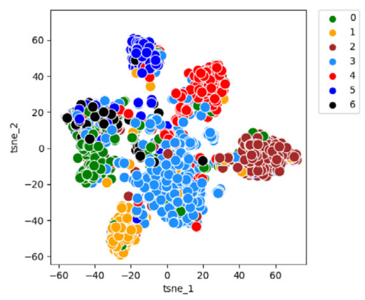
(a) Raw Features



(b) DyVGRNN

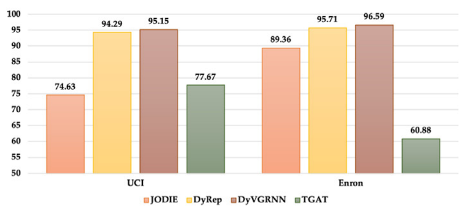


(c) SI-VGRNN

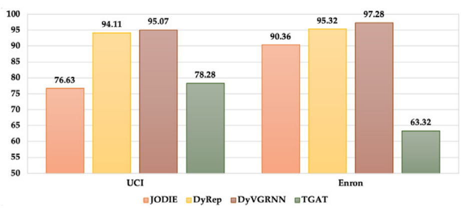


(d) DySAT

Fig. 4. Cluster visualisation for embeddings of Cora dataset in 2D space. (a) Raw feature cluster visualisation demonstrates the inability to differentiate between clusters. (b) Cluster visualisation of DyVGRNN embeddings showing distinct clusters. (c) Cluster visualisation of SI-VGRNN embeddings indicates more indiscernible clusters compared to DyVGRNN. (d) Cluster visualisation of DySAT embedding also reveals more undetectable clusters compared to the two other methods.



(a) AUC



(b) AP

Fig. 5. The comparison of the proposed DyVGRNN and continuous methods for the task of dynamic link prediction. As shown in the charts below, different methods are represented by different colours. These results are performed on UCI and Enron. Each chart shows the results for UCI and Enron in the left and right groups, respectively. (a) The results of comparing in terms of AUC score. (b) The results of comparing in terms of AP score.

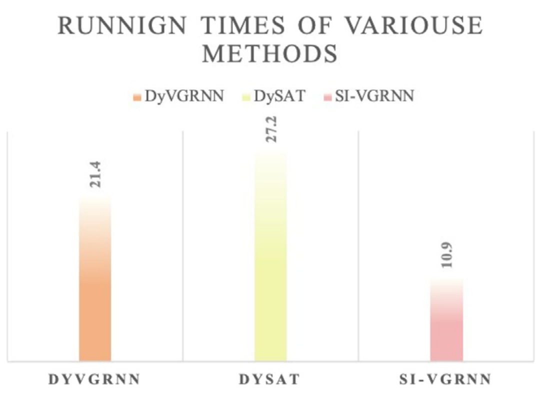


Fig. 6. Comparison of running times of different methods on the LFB dataset. The colours represent various methods in the colour scheme.

Table 7
Time complexity of different methods.

Method	Time complexity
DynAE	$O(T(\|E\| + \|V\|))$
DynRNN	$O(T\|V\|H^2)$
DynAERNN	$O(T\|V\|H^2 + T(\|E\| + \|V\|))$
HTGN	$O(T\|V\|H^2 + \|E\|H^2)$
DySAT	$O(T\|V\|H^2 + \|E\|H^2)$
VGRNN	$O(T\|V\|H^2) + O(\|E\|)$
DyVGRNN	$O(T\|V\|H^2) + O(\|E\|H^2)$

method is $O(T|V|H^2)+O(EH^2)$. Table 7 lists the time complexity of some methods evaluated in our work on LFB dataset. In addition, Fig. 6 contrasts the running times of SI-VGRNN, DySAT, and DyVGRNN. As seen, our approach runs faster than DySAT but lower than SI-VGRNN. Although compared to VGRNN, this is seen as a shortcoming for our model, accuracy at inference time is more important in many cases.

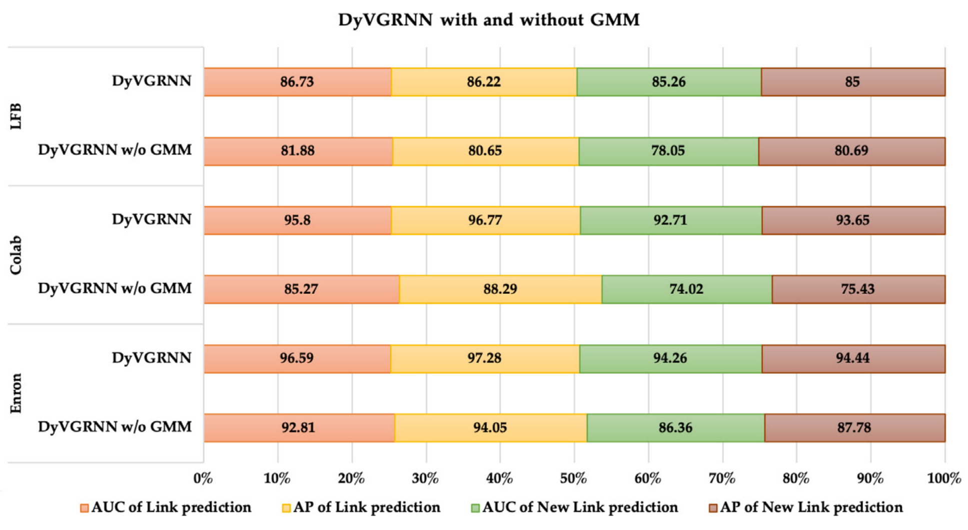


Fig. 7. The effect of GMM on proposed DyVGRNN. The outcomes of running our model in two modes with and without GMM are shown in this figure. Two tasks, dynamic link prediction, and new dynamic link prediction are performed on Enron, Colab, and Facebook with the results. The various criteria for these two tasks are represented by the colours in accordance with the colour scheme.

Table 8
Effect of parameter K on the DyVGRNN outcome. The results of dynamic link prediction and dynamic new link prediction by adjusting K to different values are given in this table. "Mean of AUCs" in each dataset category show the mean of AUC of link prediction and AUC of link prediction for different K.

Dataset	Metric	K = 2	K = 3	K = 4	K = 5	K = 6	K = 7
Enron	AUC of link prediction	95.80	96.59	95.70	96.60	95.92	95.82
	AP of link prediction	96.77	97.28	96.34	97.10	96.73	96.64
	AUC of new link prediction	92.71	94.26	93.10	93.58	93.64	92.98
	AP of new link prediction	93.65	94.44	93.36	93.38	94.25	93.57
	Mean of AUCs	94.25	95.42	94.4	95.09	94.78	94.4
	Mean of APs	95.21	95.86	94.85	95.24	95.49	95.10
Facebook	AUC of link prediction	93.17	90.20	92.55	91.37	92.61	93.02
	AP of link prediction	92.70	88.67	92.17	90.66	92.18	92.47
	AUC of new link prediction	92.51	89.79	91.95	90.70	91.93	92.55
	AP of new link prediction	91.81	88.11	91.67	89.81	91.17	92.04
	Mean of AUCs	92.84	89.99	92.25	91.03	92.27	92.78
	Mean of APs	92.25	88.39	91.92	90.23	91.67	92.25
Colab	AUC of link prediction	95.80	90.20	92.55	91.37	92.61	93.02
	AP of link prediction	96.77	88.67	92.17	90.66	92.18	92.47
	AUC of new link prediction	92.71	89.79	91.95	90.70	91.93	92.55
	AP of new link prediction	93.65	88.11	91.67	89.81	91.17	92.04
	Mean of AUCs	92.84	89.99	92.25	91.03	92.27	92.78
	Mean of APs	92.25	88.39	91.92	90.23	91.67	92.25

4.10. Ablation study

In this section, we conduct ablation studies to verify the effectiveness of the key components of the proposed model.

Selection of K

Since each dataset has various properties, we need to select the hyperparameter K according to the unique properties of each dataset. To this end, this study compares the results by examining the various values of K and selecting the best value. Table 8 shows the results of these comparisons. The best value of K for Enron, UCI, Cora, and AS733 datasets was 3, and for Facebook, Colab, and LFB datasets were 2. The first column of these tables shows the situation where the GMM does not affect the results. As can be seen, at $K = 2$, i.e. applying the GMM, a significant improvement in the results is achieved. This improvement demonstrates the validity of our claim that the use of GMM positively affects outcomes.

Impact of the GMM

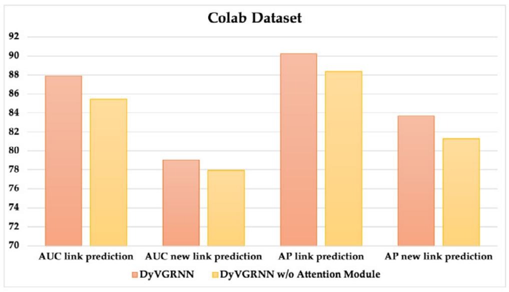
The effects of utilising a GMM to handle multimodality are examined in this section. This is accomplished by considering the proposed DyVGRNN in two different scenarios: first, without using GMM, and second, using GMM. Fig. 7 shows the result of comparisons in these two modes. As seen in this figure, GMM leads to improving the results.

Impact of the Attention Module

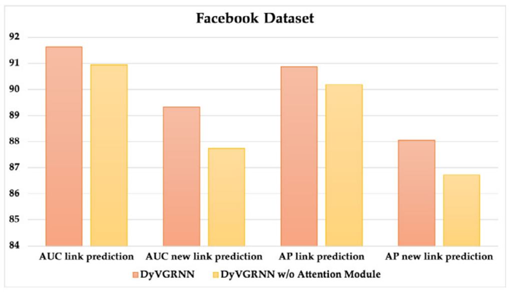
To assess the effectiveness of the attention module, we have divided the proposed model into two modes: with and without using it. To emphasise the attention module, we investigated the proposed method without considering GMM. Fig. 8 shows the result of comparisons in these two modes.

Impact of Features

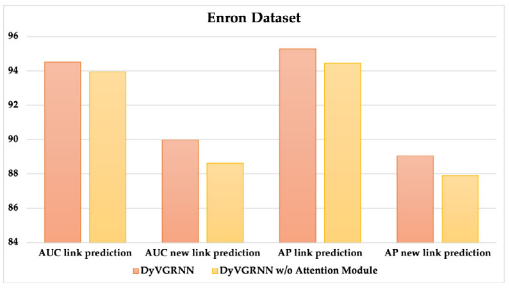
A noteworthy point in examining the results is the effect of the node features on the results. Fig. 9 shows the performance of DyVGRNN in the Cora in two modes: with and without features. The performance is significantly improved with the presence of node features, which indicates our proposed method can capture



(a) Colab Dataset

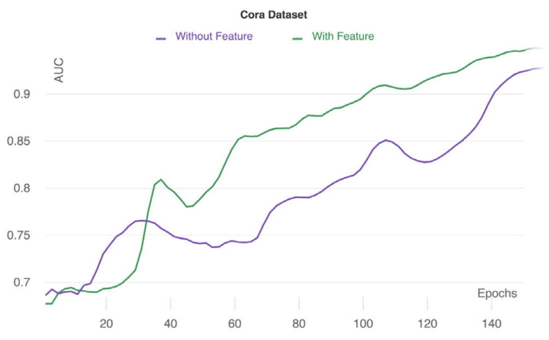


(b) Facebook Dataset

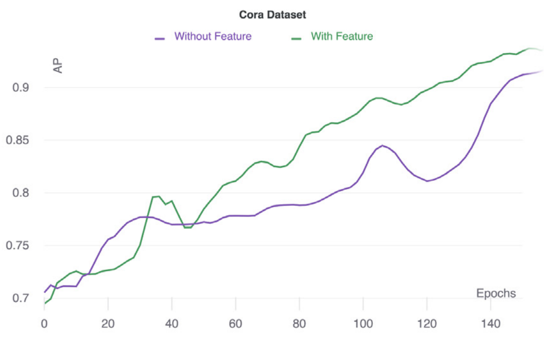


(c) Enron Dataset

Fig. 8. Impact of the attention module on DyVGRNN. In this figure, the results of running our model in two different modes with and without the attention module are depicted. To achieve this, two tasks – dynamic link prediction and new dynamic link prediction – are carried out. The colours, in accordance with the colour scheme, represent the various criteria for these two tasks. The results of the comparison on the (a) Colab, (b) Facebook, (c) Enron datasets.



(a) AUC



(b) AP

Fig. 9. The results of comparing the proposed method on Cora with and without using features. Dynamic link prediction is used to accomplish this. (a) The result of comparison in terms of AUC. Results are enhanced by the presence of node features. (b) The result of comparison in terms of AP. Results are improved when node features are present.

long-term dependencies in both the topological evolution and dynamics of node features.

5. Conclusion and future works

We proposed DyVGRNN, an integrated variational GRNN for learning node representations of dynamic graphs. DyVGRNN has additional random latent variables in the GRNN framework for capturing the evolution of graph structures and node attributes. We have shown that the combination of variational inference based on GMM and the proposed framework leads to a high level of validity and knowledge of the model. We also introduced an attention module to consider each snapshot's importance, leading to improved performance. The experiments' results showed our model's superiority over baseline and state-of-the-art methods. In the future, we are looking to apply a probabilistic decoder to the VGAE structure than a simple inner product decoder. In our proposed method, VGAEs reconstruct the adjacency matrix, but not the features matrix. Therefore, considering the reconstruction of the feature matrix and adjacency matrix would lead to a raise in accuracy. We believe it is a worthwhile area to explore more. In addition, it would intrigue to study the impact of other GNN frameworks, such as GAT, GraphSAGE, and GIN, with different layer numbers for the encoder and perhaps the decoder.

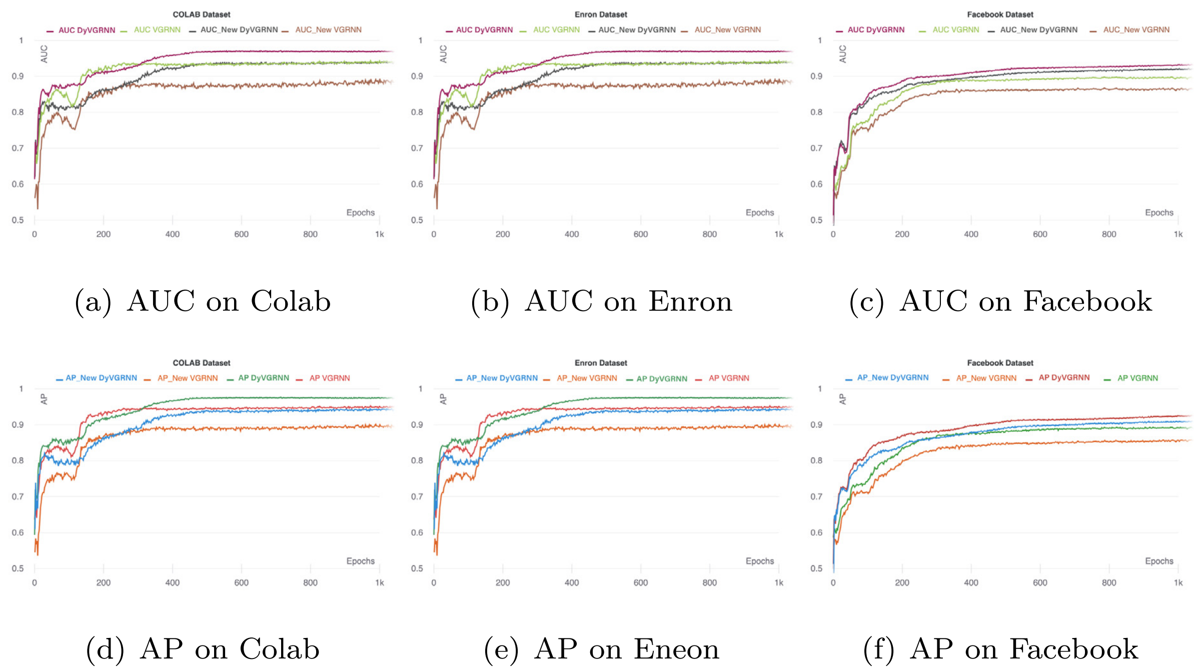


Fig. A.10. Comparing the proposed method with VGRNN on different datasets in terms of AUC and AP. The colours reflect the various criteria for dynamic link prediction and dynamic new link prediction under the colour scheme. (a) The comparison of two methods in terms of AUC on Colab. The early epochs are closely contested, but after epoch 300, DyVGRNN soon overtakes VGRNN. (b) The comparison of two methods in terms of AP on Colab. The superiority of DyVGRNN is significant after epoch 300. (c) The comparison of two methods in terms of AUC on Enron. After epoch 300, DyVGRNN's dominance becomes considerable. (d) The comparison of two methods in terms of AP on Enron. Again, in the 300th period, DyVGRNN's advantage becomes substantial. (e) The comparison of two methods in terms of AUC on Facebook. Even in the early epochs, DyVGRNN's supremacy was noticeable. (f) The comparison of two methods in terms of AP on Facebook. From the very beginning, DyVGRNN's dominance is significant.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

DAC was supported by an NIHR Research Professorship, an RAEng Research Chair, the InnoHK Hong Kong Centre for Cerebro-cardiovascular Health Engineering (COCHE), the NIHR Oxford Biomedical Research Centre (BRC), and the Pandemic Sciences Institute at the University of Oxford.

Appendix A. Visualisation the results of comparison

In order to more thoroughly assess the performance of the proposed method, DyVGRNN and VGRNN are compared in Fig. A.10. It is evident that DyVRNN performs better over time in practically all epochs. Despite having close competition in the early epochs, DyVGRNN quickly passes VGRNN and establishes its superiority.

Appendix B. Qualitative analysis

Fig. B.11 displays a visualisation of the learnt embeddings over time to show how effectively the embeddings are encoded. To do so, we use the clustering task and the silhouette metric on synthetic data and visualised the learnt embeddings in a two-dimensional space throughout our training. The clusters become more well-separated with time, as can be observed.

Appendix C. Node classification task

We compare our model to three baseline methods in order to assess its performance on the classification task. Two of these methods, GCN (Kipf & Welling, 2017) and GraphSAGE (Hamilton, Ying, & Leskovec, 2017a), are supervised techniques that relied solely on static graph structures and node attributes, ignoring temporal information. Another method, RNNGCN (Yao & Joe-Wong, 2021), utilised a two-layer GCN with a decay weight as a learnable parameter. This decay weight has applied to information from each timestep, gradually decreasing over time. The resulting linear combination of information over time is then used for classification purposes.

The datasets used in this task has obtained from DBLP,[3] a comprehensive database of academic papers in various subfields of computer science. The authors of these papers are represented

[3] https://dblp.org/.

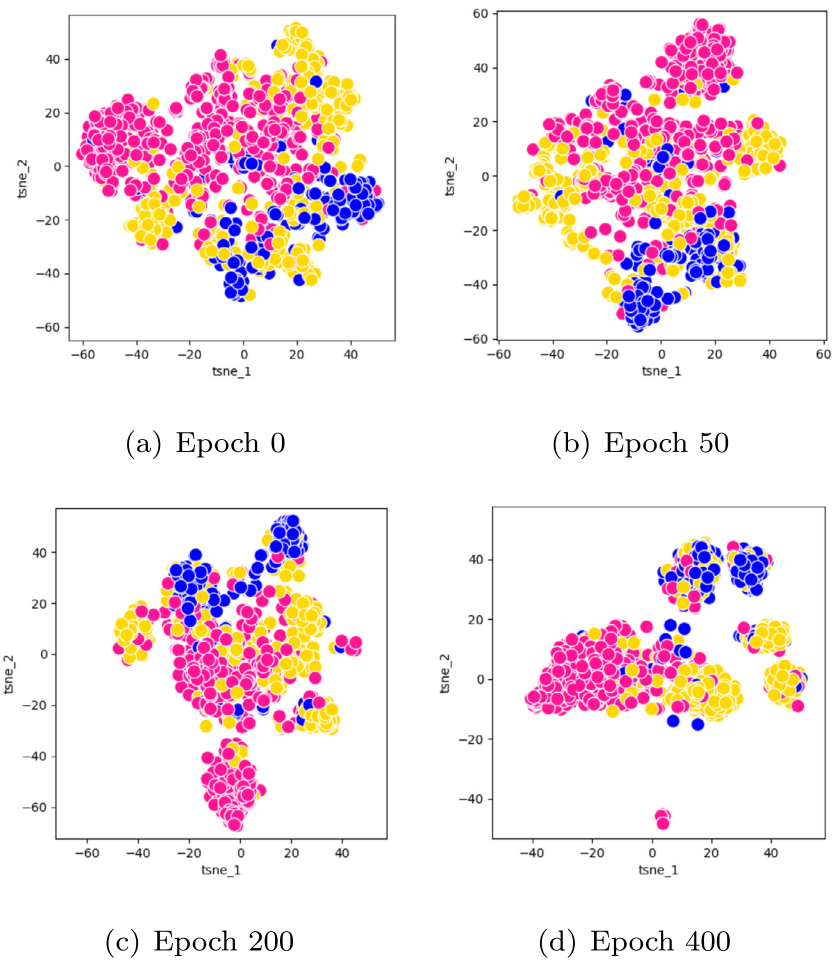


Fig. B.11. Visualisation of the learnt embeddings of DyVGRNN over time. In this figure, each colour corresponds to a cluster. (a) Visualisation of embedding on epoch 0 of the running. The clusters are confused. (b) Visualisation of embedding on epoch 50 of the running. Clusters are hardly distinguishable. (c) Visualisation of embedding on epoch 200 of the running. Clusters show themselves, but they are still intertwined. (d) Visualisation of embedding on epoch 400 of the running. Clusters are almost easily distinguishable.

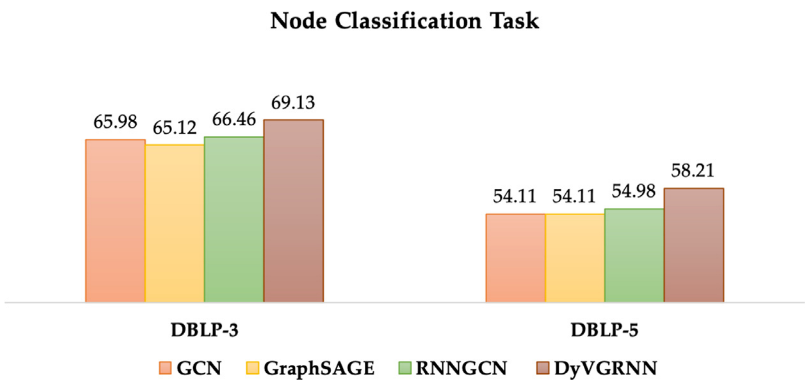


Fig. C.12. The results of comparing the classification performance of the proposed method on DBLP-3 and DBLP-5 datasets with other baselines in terms of AUC. The colours, in accordance with the colour scheme, represent the various methods.

as nodes in a graph, with connections between nodes indicating co-authorship. Analysing the authorship of papers published between 2005 and 2018 resulted in the dynamic graph in these datasets, treating each year as a snapshot. DBLP-5 has 6606 nodes, 42 815 edges, and 10 snapshots, while DBLP-3 has 4257 nodes, 23 540 edges, and 10 snapshots. These datasets included node attributes extracted by word2vec (Mikolov, Chen, Corrado, & Dean, 2013) from authors' paper titles and abstracts. They both have 100 attributes. These datasets are further clustered into three and five classes, respectively, based on the research area of the authors. These classes remained static over time. Fig. C.12 shows the results of our comparison in terms of the AUC. As seen, our proposed DyVGRNN outperforms other methods in both datasets.

References

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on world wide web* (pp. 37–48).

Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, *40*(1), 1–39.

Bacciu, D., Errica, F., Micheli, A., & Podda, M. (2020). A gentle introduction to deep learning for graphs. *Neural Networks*, *129*, 203–221.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR* (pp. 1–15).

Cao, S., Lu, W., & Xu, Q. (2015). Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 891–900).

Charte, D., Charte, F., del Jesus, M. J., & Herrera, F. (2020). An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges. *Neurocomputing*, *404*, 93–107.

Chen, J., Wang, X., & Xu, X. (2022). GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence*, *52*(7), 7513–7528.

Chen, J., & Zhang, A. (2020). Hgmf: heterogeneous graph-based fusion for multimodal data with incompleteness. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1295–1305).

Chen, J., Zhang, J., Xu, X., Fu, C., Zhang, D., Zhang, Q., et al. (2019). E-lstm-d: A deep learning framework for dynamic network link prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(6), 3699–3712.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th international conference on neural information processing systems* (pp. 3844–3852).

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., et al. (2017). Deep unsupervised clustering with gaussian mixture variational autoencoders. In *ICLR* (pp. 1–12).

Fout, A., Byrd, J., Shariat, B., & Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 6533–6542).

Gao, J., & Ribeiro, B. (2022). On the equivalence between temporal and static equivariant graph representations. In *International conference on machine learning* (pp. 7052–7076). PMLR.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, *3*(Aug), 115–143.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goyal, P., Chhetri, S. R., & Canedo, A. (2020). dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, *187*, Article 104816.

Goyal, P., Kamra, N., He, X., & Liu, Y. (2018). DynGEM: Deep embedding method for dynamic graphs. CoRR abs/1805.11273. URL http://arxiv.org/abs/1805.11273.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855–864).

Hajiramezanali, E., Hasanzadeh, A., Duffield, N., Narayanan, K., Zhou, M., & Qian, X. (2019). Variational graph recurrent neural networks. In *Proceedings of the 33rd international conference on neural information processing systems* (pp. 10701–10711).

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, *14*(3), 1–159.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 1025–1035).

Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *IEEE Data(base)*, 1–24.

Han, Z., Jiang, J., Wang, Y., Ma, Y., & Tresp, V. (2019). The graph hawkes network for reasoning on temporal knowledge graphs. In *Learning with temporal point processes workshop at at the 33rd conference on neural information processing systems (NeurIPS 2019) NeurIPS 2019*.

Jin, W., Qu, M., Jin, X., & Ren, X. (2020). Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the 2020 conference on empirical methods in natural language processing* EMNLP, (pp. 6669–6683).

Ju, W., Luo, X., Ma, Z., Yang, J., Deng, M., & Zhang, M. (2022). GHNN: Graph Harmonic Neural Networks for semi-supervised graph-level classification. *Neural Networks*, *151*, 70–79.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)* (pp. 1–15).

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *ICLR* (pp. 14–16).

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., & Zemel, R. (2018). Neural relational inference for interacting systems. *Proceedings of Machine Learning Research*, *80*, 2688–2697.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. In *NIPS workshop on bayesian deep learning* (pp. 1–12).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR* (pp. 1–14).

Kostantinos, N. (2000). Gaussian mixtures and their applications to signal processing. In *Advanced signal processing handbook: Theory and implementation for radar, sonar, and medical imaging real time systems*. CRC Press, 3–1.

Kumar, S., Zhang, X., & Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1269–1278).

Lei, K., Qin, M., Bai, B., Zhang, G., & Yang, M. (2019). GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE INFOCOM 2019-IEEE conference on computer communications* (pp. 388–396). IEEE.

Leskovec, J., Kleinberg, J., & Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 177–187).

Li, J., Han, Z., Cheng, H., Su, J., Wang, P., Zhang, J., et al. (2019). Predicting path failure in time-evolving graphs. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1279–1289).

Liu, X., Hsieh, P.-C., Duffield, N., Chen, R., Xie, M., & Wen, X. (2019). Real-time streaming graph embedding through local actions. In *Companion proceedings of the 2019 world wide web conference* (pp. 285–293).

Liu, Y., Shi, X., Pierce, L., & Ren, X. (2019). Characterizing and forecasting user engagement with in-app action graph: A case study of snapchat. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2023–2031).

Ma, Y., Guo, Z., Ren, Z., Tang, J., & Yin, D. (2020). Streaming graph neural networks. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 719–728).

Manessi, F., Rozza, A., & Manzo, M. (2020). Dynamic graph convolutional networks. *Pattern Recognition*, *97*, Article 107000.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of workshop at ICLR*.

Molaei, S., Bousejin, N. G., Zare, H., & Jalili, M. (2021). Deep node clustering based on mutual information maximization. *Neurocomputing*, *455*, 274–282.

Molaei, S., Bousejin, N. G., Zare, H., Jalili, M., & Pan, S. (2021). Learning graph representations with maximal cliques. *IEEE Transactions on Neural Networks and Learning Systems*, 1–8.

Mudiyanselage, T. B., Lei, X., Senanayake, N., Zhang, Y., & Pan, Y. (2022). Predicting CircRNA disease associations using novel node classification and link prediction models on Graph Convolutional Networks. *Methods*, *198*, 32–44.

Narayan, A., & Roe, P. H. (2018). Learning graph dynamics using deep neural networks. *IFAC-PapersOnLine*, *51*(2), 433–438.

Niknam, G., Molaei, S., Zare, H., Clifton, D., & Pan, S. (2022). Graph representation learning based on deep generative Gaussian mixture models. *Neurocomputing*, 157–169.

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1105–1114).

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., & Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 2609–2615).

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., et al. (2020). Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 34* (pp. 5363–5370).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701–710).

Priebe, C. E., Conroy, J. M., Marchette, D. J., & Park, Y. (2005). Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, *11*(3), 229–247.

Rahman, M., & Al Hasan, M. (2016). Link prediction in dynamic networks using graphlet. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 394–409). Springer.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning* (pp. 1278–1286). PMLR.

Salha-Galvan, G., Lutzeyer, J. F., Dasoulas, G., Hennequin, R., & Vazirgiannis, M. (2022). Modularity-aware graph autoencoders for joint community detection and link prediction. *Neural Networks*, 474–495.

Sankar, A., Wu, Y., Gou, L., Zhang, W., & Yang, H. (2020). Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining* (pp. 519–527).

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93.

Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing* (pp. 362–373). Springer.

Shen, T., Jiang, J., Zhou, T., Pan, S., Long, G., & Zhang, C. (2018). Disan: directional self-attention network for RNN/CNN-free language understanding. In *Proceedings of the thirty-second AAAI conference on artificial intelligence and thirtieth innovative applications of artificial intelligence conference and eighth AAAI symposium on educational advances in artificial intelligence* (pp. 5446–5455).

Skarding, J., Gabrys, B., & Musial, K. (2021). Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, *9*, 79143–79168.

Taheri, A., Gimpel, K., & Berger-Wolf, T. (2019). Learning to represent the evolution of dynamic graphs with recurrent models. In *Companion proceedings of the 2019 world wide web conference* (pp. 301–307).

Tan, Z., Wang, M., Xie, J., Chen, Y., & Shi, X. (2018). Deep semantic role labeling with self-attention. In *Proceedings of the thirty-second AAAI conference on artificial intelligence and thirtieth innovative applications of artificial intelligence conference and eighth AAAI symposium on educational advances in artificial intelligence* (pp. 4929–4936).

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 6000–6010).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations* (pp. 1–18).

Viswanath, B., Mislove, A., Cha, M., & Gummadi, K. P. (2009). On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks* (pp. 37–42).

Wang, Y., Chang, Y.-Y., Liu, Y., Leskovec, J., & Li, P. (2020). Inductive representation learning in temporal networks via causal anonymous walks. In *International conference on learning representations*.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Xu, K. S., & Hero, A. O. (2014). Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, *8*(4), 552–562.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. In *ICLR* (pp. 1–12).

Yang, M., Zhou, M., Kalander, M., Huang, Z., & King, I. (2021). Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 1975–1985).

Yao, Y., & Joe-Wong, C. (2021). Interpretable clustering on dynamic graphs with recurrent graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 35* (pp. 4608–4616).

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., et al. (2019). T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, *21*(9), 3848–3858.