A Dynamic Variational Framework for Open-World Node Classification in Structured Sequences

Qin Zhang
Shenzhen University
Guangdong, China
qinzhang@szu.edu.cn

Qincai Li
Shenzhen University
Guangdong, China
liqincai2020@email.szu.edu.cn

Xiaojun Chen*
Shenzhen University
Guangdong, China
xjchen@szu.edu.cn

Peng Zhang
Guangzhou University
Guangdong, China
p.zhang@gzhu.edu.cn

Shirui Pan
Griffith University
Queensland, Australia
s.pan@griffith.edu.au

Philippe Fournier-Viger
Shenzhen University
Guangdong, China
philfv@qq.com

Joshua Zhexue Huang
Shenzhen University
Guangdong, China
zx.huang@szu.edu.cn

***Abstract*—Structured sequences are a popular data representation, used to model complex data such as traffic networks. A key machine learning task for structured sequences is node classification, that is predicting the class labels of unlabeled nodes. Though many node classification models were proposed, they assume a *closed world* setting, that all class labels appear in the training data. But in the real-world, the presence of never-before-seen class labels in testing data can considerably degrade a classifier's accuracy. A promising solution to this issue is to build classifiers for an open-world setting, where samples with unknown class labels are continuously observed such that training and testing data may have different class label spaces. Several approaches have been proposed for open-world learning problems in computer vision and natural language processing, but they cannot be applied directly to structured sequences due to the complexity of their non-Euclidean properties and their dynamic nature. This paper addresses this important research gap by proposing a novel Open-world Structured Sequence node Classification (OSSC) model, to learn from structured sequences in an open-world setting. OSSC captures the structural and temporal information via a GCN-based dynamic variational framework. A latent distribution sequence is learned for each node using both stochastic states and deterministic states, to capture the evolution of node attributes and topology, followed by a sampling process to generate node representations. An open-world classification loss is further adopted to ensure that node representations are sensitive to unknown classes. And a combination of *Openmax* and *Softmax* is utilized to recognize nodes from unknown classes and to classify others to one of the known classes. Experiments on real-world datasets show that the proposed OSSC method is capable of learning accurate open-world node classifiers from structured sequence data.**

***Index Terms*—Open-world learning, Structured sequences, Dynamic variational autoencoder**

I. Introduction

Machine learning from structured data plays a key role in numerous fields. To represent complex relationships between objects and how they evolve over time, a popular data model is structured sequences. A *structured sequence* (also known as dynamic [1], evolving [2], time-varying [3], [4], or temporal [5] graphs/networks) may be viewed as a sequence $G_1, G_2, \ldots, G_T$ of static attributed graphs observed at different timestamps. Research on structured sequences is essential as real-world systems are generally dynamic, and evolve rapidly over time [6]. For instance, structured sequences are well-suited to model changes in traffic networks [7], social networks [8], and proteins [9].

A key task in structured sequence learning is *node classification*, that is to predict the labels of unlabeled nodes, and has many applications such as traffic state prediction [10], and disease classification [11]. Node classification in a structured sequence is difficult because it requires considering not only the influence of node labels and the topology but also the time. Besides, a general challenge for machine learning with structured sequences is that unlike image or speech data, structured sequences cannot be easily represented in Euclidean space to benefit from its properties.

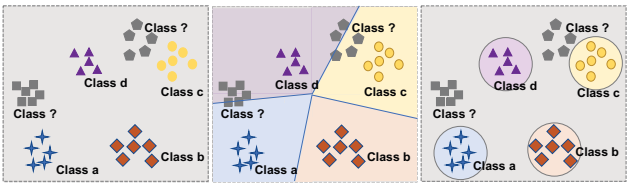

Fig. 1: Comparison between closed-world classification and open-world classification. The left figure depicts an original dataset distribution with four known classes (*class a,b,c,d*) and two unknown classes (*class ?*). The middle figure shows the decision boundary of each class obtained by a traditional closed-world classification method, which will obviously misclassify unknown class samples into known classes during testing. The right figure depicts an open-world classifier, where decision boundaries limit the scope of the known classes *a,b,c,d*, keeping space for unknown classes. Via these decision boundaries, samples from unknown classes will be labeled as "unknown" rather than misclassified as one of known classes.

Recently, several machine learning frameworks such as STAR [6] and Aspen [12] were proposed to facilitate struc-

*Corresponding author.

2374-8486/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDM54844.2022.00081

tured sequence data analysis. However, current models for handling structured sequence data implicitly assume that samples are collected from a closed-world setting [13], [14] where all streaming examples are from a static class label space [15]–[17]. For example, some methods for graph node classification [6], [18], [19] that strictly follow the closed-world assumption are SS-GSELM [19], which combines dynamic graph learning with self-paced learning in a semi-supervised scenario, and STAR [6], which uses a spatiotemporal attentive recurrent network to extract the vector representation of the neighborhood by sampling and aggregating local neighbor nodes. Even though these methods achieve satisfactory results in the closed-world learning scenario, they cannot handle unknown class labels in the open-world learning scenario because these graph models will incorrectly classify unknown-class samples into one of the known classes [13], [20], [21]. Considering the needs of practical applications, open-world learning [22] was proposed where new samples may come from new label spaces, and open-world classifiers are required to solve the challenges of both known-class classification and unknown-class recognition.

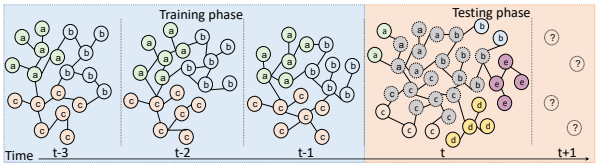

Fig. 2: An example of open-world learning from a structured sequence. In the training phase, a sequence of graphs is observed, having a dynamic structure and features, where nodes belong to three classes, i.e., *class a*, *b*, and *c*. Data from the testing domain is inconsistent with that of the training domain in terms of class labels. The class label space for testing consists of elements having unknown class labels (such as *class d* and *class e*), which were not seen in training.

Open-world learning models require strong generalization capability [23], [24] to handle an unknown class label space. It has received increasing attention from researchers in computer vision (CV) and natural language processing (NLP) [25] [26]. In computer vision, ORE [23] adopts contrastive clustering and energy based unknown identification for open-world object detection. Perera *et al.* [27] formalized this as an *open set recognition* problem, and trains a generative model for all known classes. In natural language processing, Zheng *et al.* [28] formalized open-world learning as an out-of-domain detection problem. Yan *et al.* [29] used a semantic-enhanced Gaussian mixture (SEG) model for unknown intent detection. In structured data analysis, OpenWGL [22] [30] was proposed for static graph open-world learning. Another study [31] presented an incremental training method for open-world lifelong learning on graphs which focuses on the catastrophic forgetting and cold start problems. Building on these foundations, this work attempts to address the open-world learning problem for structured sequences, where we focus on the problem of accurately identifying samples of unknown classes in the testing phase.

The input data for this problem is a structured sequence, where the testing data contain both known-class nodes and unknown-class nodes. Open-world learning on a structure sequence aims to learn a new classifier that either classifies a new node into one of the known classes or rejects it from all the known classes. In order to achieve this goal, there are numerous challenges:

- *Challenge 1*: Learning approximate representations (features) among nodes from same classes and establish relatively closed class boundaries, to keep space for unknown classes. As shown in Fig. 1, traditional closed-world classification methods normally adopt open-end classification boundaries and will misclassify unknown classes into known classes during testing. For open-world learning, we need closed-end class boundaries and compact categories in the learned feature space.
- *Challenge 2*: Learning an open-world classifier that can solve the class label inconsistency problem between the training and testing domains of structured sequence data. As shown in Fig. 2, in the training domain, the class label space consists of instances of *classes a, b, and c*. However, in the testing domain, instances of unknown classes are encountered (such as *class d* and *e*). An open-world classifier must classify data samples either into the already known class label space, or the unknown-class label space.

In light of the above challenges, we present a new Open-world Structural Sequence node Classification model (OSSC for short) for learning from structured sequences. To solve Challenge 1, OSSC proposes a graph convolutional network (GCN)-based dynamic variational framework that captures both structural and temporal dynamics of graph sequences. It learns a latent distribution sequence for each node via sampling to generate node representations that are more stable. The adoption of an open-world classification loss further ensure that a node representation is closely related to its own label, to yield more compact categories and making the sample sensitive to unknown classes. To solve challenge 2, Besides the open-world classification loss, a combination of *Openmax* and *Softmax* is utilized to recognize nodes from unknown classes and to classify others to one of the known classes. The contributions of the paper are summarized as follows:

- A new Open-world Structural Sequence node Classification model is proposed to solve the problem of open-world learning from structured sequences.
- OSSC solves the technical challenges to jointly capture temporal and structural dynamics, and a time-changing class label space from open-world structured sequences. To learn a node representation and classification model from structured sequences, OSSC uses a dynamic graph variational framework to capture the temporal and structural dynamics. Besides using commonly used deterministic states, OSSC relies on stochastic states to learn

a latent distribution for each node at each timestamp. dynamic variational autoencoder (DVAE) loss and open-world classification loss are utilized to restrict model learning and *Openmax* and *Softmax* are used to obtain the final classification and recognition function.
- Experiments on real-world structured sequences have shown that the proposed method is capable of generating accurate node classifiers from structured sequences.

The rest of the paper is organized as follows. Section II defines the problem and the proposed learning framework. Section III introduces the proposed method in detail. Section IV presents the algorithm. Section V report results from experiments. Lastly, Section VI draws a conclusion and discusses future work.

II. Preliminaries

This section introduces the problem definition, and then the overall framework of the proposed OSSC method.

A. Problem Description

Consider static structured data denoted as a graph $G = (V, E, A, X, Y)$, where $V = \{v_i\}_{i=1,\dots,N}$ is a set of N nodes in the graph, and $e_{i,j} = (v_i, v_j) \in E$ is an edge indicating the relationship between two nodes. The topological structure of G is represented by an adjacency matrix A, where $A_{i,j} = 1$ if $(v_i, v_j) \in E$; otherwise $A_{i,j} = 0$. $x_i \in X$ indicates features associated with each node v_i. $Y \in \mathbb{R}^{N \times C}$ is a label matrix of G, where C is the already-known node categories (classes). If a node $v_i \in V$ is associated with a label c, $Y^c_{(i)} = 1$; otherwise, $Y^c_{(i)} = 0$.

A structured sequence $\mathbb{G}$ can be denoted as a sequence of static graphs $\mathbb{G} = \{G_1, \dots, G_T\}$, where $G_t = (V_t, E_t, A_t, X_t, Y_t)$ represents a graph observed at time t. Then, the task of *node classification for a structured sequence* can be described as follows: given a structured sequence $\mathbb{G}$, the aim is to learn the label $y^{(t)}_{(i)}$ of each node v_i at each timestamp t of the sequence. We aim to learn a mapping function of $f_t : \{G_1, \dots, G_T\} \to Y \in \mathbb{R}^{N \times C}$, i.e., $y^{(t)} = f_t(G_1, \dots, G_t)$, such that $y^{(t)}$ can capture temporal patterns required to optimize $y^{(t+1)}$. In many scenarios, each node's class label remains unchanged in a sequence, and $y^{(t)}_{(i)}$ can be simplified as $y_{(i)}$. The final prediction $Y = y_{(i)} = f_T(G_1, \dots, G_T), i = 1, \dots, N$ is also expected.

Now, we formally describe the problem of **open-world learning on structured sequences.** Let there be a structured sequence $\mathbb{G} = \{G_1, \dots, G_T\}$, $G_t = (V_t, E_t, A_t, X_t, Y_t)$, $V_t = V_t^{train} \cup V_t^{test}$, where V_t^{train} denotes training nodes with class labels obtained at time t and V_t^{test} denotes testing data without class labels. Furthermore, let $V_t^{test} = \mathbb{S} \cup \mathbb{U}$, where $\mathbb{S}$ is a set of nodes belonging to the already known classes whose class label has already appeared in V_t^{train}, and $\mathbb{U}$ is a set of nodes not belonging to the already-known classes whose class label has never been observed before. Then, open-world learning on structured sequences aims to learn a $(C+1)$-$class$ classifier, where $f(V_T^{test}) \to Y, (Y \in \{1, \dots, C, unknown\})$ either classifies each node v to one of the already known classes, or rejects it from all the known classes, taking it as a sample from an unknown class.

B. Overall Framework

Fig. 3 shows the framework of OSSC. It consists of three main components. The first one learns temporal and structural information from graph sequences. The second component generates node representations, and the third component trains an open-world learning classifier.

The purpose of the first component (**temporal and structure learning**) is to capture the dynamic information in a structured sequence. Many graph neural network models only generate deterministic representations for nodes, and thus they cannot model the uncertainty in a stream. Based on the idea of Variational Graph Autoencoder network (VGAE), OSSC learns a sequence of latent distributions of each node, so as to learn both deterministic latent variables and stochastic latent variables. In this way, OSSC can simultaneously capture the evolution of structured sequences and learn the final latent distribution of each node to obtain more stable representations.

In the second component of **node representation generation**, based on the latest learned distribution, OSSC generates M types of representations for each node by sampling from the truncated Gaussian distribution, where the samples with tail-probability are removed.

In the third component of **open-world classifier learning,** to learn a good classifier for known classes and an effective detector for unknown classes, OSSC introduces an open-world classification loss to describe whether a node belongs to an existing class or an unknown class. A sampling process generates multiple versions of feature vectors to test the certainty that a node belongs to known classes, while *openmax* and *softmax* are utilized to reject nodes from unknown classes and classify others in known classes.

III. The proposed model

OSSC consists of three components, i.e., temporal and structural learning, node representation generation, and open-world classifier learning. All the three types of learning are combined to formulate the ultimate learning function.

A. Temporal and structural learning

Given a structured sequence, encoding latent features for each node requires to take into consideration the uncertainty of the node. For this, we combine a graph convolutional network (GCN) with a Dynamical Variational AutoEncoder (DVAE) network to generate a latent distribution for each node.

Given a structured sequence $\mathbb{G} = \{G_1, \dots, G_T\}$, consider a specific graph snapshot taken at timestamp t, i.e., $G_t = (V_t, E_t, A_t, X_t)$. To learn node feature X_t and graph structure A_t, OSSC first uses a GCN [32] to learn the representation of each node based on feature information propagation in the graph.

The first GCN layer generates a lower-dimensional feature matrix at timestamp t as follows,

$$H_t^{(1)} = GCN(X_t, A_t) = ReLU(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}} X_t W^{(1)}), \quad (1)$$

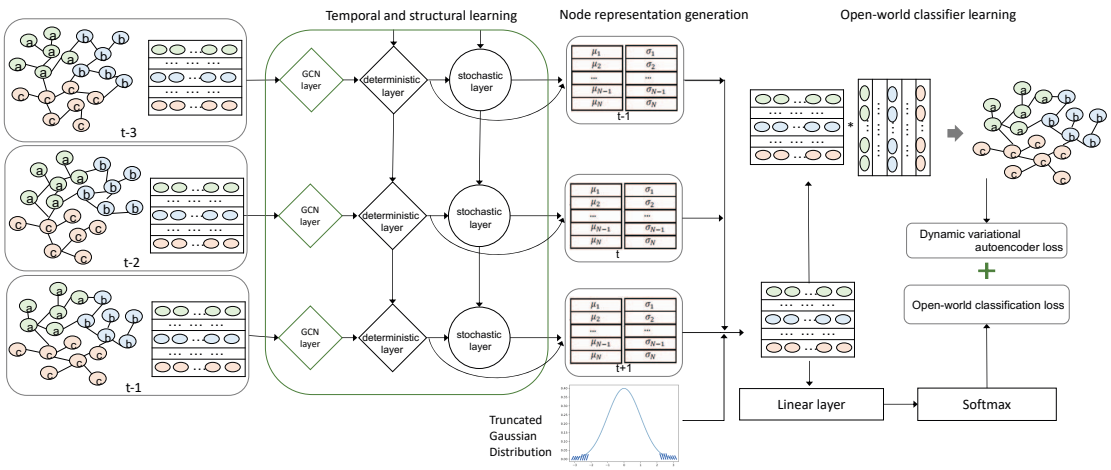

Fig. 3: The framework of the proposed Open-world Structural Sequence node Classification (OSSC) method. The input is a sequence of graphs where each node has structure and attribute information. The training process includes three main components, that are temporal and structural learning, node representation generation, and open-world classifier learning. The model is trained with two types of loss functions, i.e., the dynamic variational autoencoder loss which includes the KL divergence loss and the reconstruction loss, and the open-world classification loss. As a result, OSSC can learn node representations generated from a truncated Gaussian distribution, which is sensitive to unknown classes. More details are given in Section III.

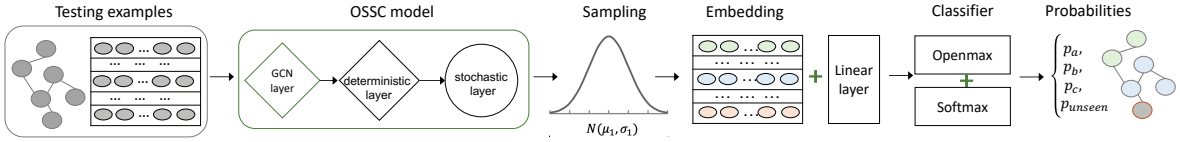

Fig. 4: The classification and identification process. For nodes in the testing set, the OSSC model generates feature embedding from each node by sampling. The linear layer reduces the dimensionality to the corresponding number of categories, then *openmax* is utilized to classify a sample into the unknown-class and softmax is applied to identify a known-class.

where $\tilde{A} = A + I_n$ is an adjacency matrix with self-loops, $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$ and $\tilde{D}_{i,j} = 0$ if $i \neq j$. Accordingly, $\tilde{D}^{\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}$ is the normalized adjacency matrix. $W^{(1)}$ is the trainable parameters of the GCN network, and $ReLU(\cdot)$ denotes the activation function.

At the second layer, we calculate the deterministic states d_t, which is determined through the recursion

$$d_t = f_{\theta_d}(d_{t-1}, H_t^{(1)}), \tag{2}$$

where f_{θ_d} is a recurrent neural network (RNN) with parameter θ_d. Specifically, we use a long short-term memory (LSTM) network in experiments. For simplicity, we use a general formulation of RNN as follows,

$$\begin{aligned} d_t = H_t^{(2)} &= RNN(H_{t-1}^{(2)}, H_t^{(1)}) \\ &= \tanh(\overrightarrow{W}^{(2)} H_t^{(1)} + \overrightarrow{U}^{(2)} H_{t-1}^{(2)} + \overrightarrow{b}^{(2)}), \end{aligned} \tag{3}$$

where $\theta_d = \{\overrightarrow{W}^{(2)}, \overrightarrow{U}^{(2)}, \overrightarrow{b}^{(2)}\}$ is the parameters of the RNN.

Moreover, a separated stochastic layer is added to learn the stochastic state z_t in the iterative process for the nodes. Following the structure of DVAE, the posterior inference of z_t is conducted by using an inference network, which uses a backward-recurrent state a_t to approximate the nonlinear dependence of z_t on future observations:

$$a_t = g_{\theta_a}(a_{t+1}, [d_t, X_t]). \tag{4}$$

To be specific, a_t can be calculated as

$$\begin{aligned} a_t = H_t^{(3)} &= RNN(H_t^{(2)}, H_{t+1}^{(3)}, X_t) \\ &= \tanh([H_t^{(2)}, X_t]\overleftarrow{W}^{(3)} + H_{t+1}^{(3)}\overleftarrow{U}^{(3)} + \overleftarrow{b}^{(3)}), \end{aligned} \tag{5}$$

where $d_t = H_t^{(2)}$ is the deterministic state, X_t is the node original feature matrix, and $\theta_a = \{\overleftarrow{W}^{(3)}, \overleftarrow{U}^{(3)}, \overleftarrow{b}^{(3)}\}$ is the backward-part parameters of the RNN.

Finally, the fourth layer models the stochastic state z_t, and obtains the latent distribution of each node. We let z_t be continuous and follow a multivariate Gaussian distribution. According to the inference model [33], we have,

$$q(z_t|z_{t-1}, a_t) = \prod_{i=1}^{N} q(z_t^i|z_{t-1}, a_t) \tag{6}$$

Furthermore, embedding the above equation into a stochastic recurrent neural network structure leads to

$$z_t = f_{\theta_z}(z_{t-1}, a_t), \tag{7}$$

i.e.,

$$\begin{aligned} H_t^{(4)} &= [\mu_t, \sigma_t] = RNN(H_t^{(3)}, H_{t-1}^{(4)}) \\ &= \tanh(H_t^{(3)}\overrightarrow{W}^{(4)} + H_{t-1}^{(4)}\overrightarrow{U}^{(4)} + \overrightarrow{b}^{(4)}) \end{aligned} \tag{8}$$

Then, we have

$$q(z_i|z_{t-1}, a_t) = \mathbb{G}(z_i|\mu_t^i, diag((\sigma_t^i))) \tag{9}$$

where μ_t is the matrix of mean vector μ_t^i at timestamp t. σ_t is the standard variance matrix of the distribution at timestamp t. Then, we can calculate the representation of nodes r_t, using a parameterization trick [33] as follows,

$$z_t = \mu_t + \sigma_t \cdot \zeta, \zeta \sim \mathbb{G}(\mathbf{0}, \mathbf{1}), \tag{10}$$

where $\mathbf{0}$ is a vector of zeros and $\mathbf{1}$ is the identity matrix. By using the temporal latent variable $z_{1:T}$, OSSC can capture the data dynamics and structural patterns.

B. Node representation generation

Following the idea of VAE methods that use a Gaussian distribution to mimic the original distribution of a latent vector, we use the Gaussian distribution to describe the latent features of each node. By sampling, we generate M different versions of feature vectors $(z_i^1, \ldots, z_i^M)$ for each node v_i from the corresponding distribution based on Eq. (10), which is also known as a reparametrization trick. In addition, during the sampling process, in order to enhance the robustness, we discard tail probabilities [34], which means that when a sampling is outside the range $[-2\sigma, 2\sigma]$, it will be discarded and resampled. σ is the corresponding standard deviation. Thus, the proposed method can be more stable for node representation learning while capturing the evolution information of a dynamic structured sequence.

C. Open-world node classifier learning

When obtaining the latent variable z_t at each timestamp $t = 1 : T$, we use a decoder model to reconstruct the graph structure A_t to learn the relationship between two nodes. The graph decoder model is defined by a generative model [33] as follows,

$$\begin{aligned} &P(A_t|z_t, d_t) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_t^{i,j}|z_t^i, z_t^j, d_t^i, d_t^j) \\ &p(A_t^{ij} = 1|z_t^i, z_t^j, d_t^i, d_t^j) = f_s((z_t^i)^T z_t^j + \lambda(d_t^i)^T d_t^j), \end{aligned} \tag{11}$$

where A_t^{ij} are the elements of A_t, and f_s denotes the logistic sigmoid function.

For the ultimate learning function, we use two loss functions, i.e., the DVAE loss, $\mathscr{L}_{DVAE}$, and the open-world classification loss, $\mathscr{L}_{OWC}$, to optimize the model, i.e.,

$$\mathscr{L}_{total} \mapsto \mathscr{L}_{DVAE} + \mathscr{L}_{OWC}. \tag{12}$$

To learn class discriminative node representations, we optimize the dynamic variational graph autoencoder module by maximizing a variational evidence lower bound (ELBO) [35], which is also called the DVAE loss,

$$\begin{aligned} \mathscr{L}_{DVAE} =& \sum_{i=1}^{N}\sum_{t=1}^{T} \mathbb{E}_{q_\phi^*(z_{t-1})}[\mathbb{E}_{q_\phi(z_t|z_{t-1}, \tilde{d}_{t:T}, x_{t:T})}[\log p_\theta(x_t|z_t, \tilde{d}_t)] \\ &- KL(q_\phi(z_t|z_{t-1}, \tilde{d}_{t:T}, x_{t:T})||p_\theta(z_t|z_{t-1}, \tilde{d}_t))] \\ =& \sum_{i=1}^{N}\sum_{t=1}^{T} \mathbb{E}_{q_\phi^*(z_{t-1})}[\mathbb{E}_{q_\phi(z_t|z_{t-1}, a_t)}[\log p_\theta(x_t|z_t, \tilde{d}_t)] \\ &- KL(q_\phi(z_t|z_{t-1}, a_t)||p_\theta(z_t|z_{t-1}, \tilde{d}_t))], \end{aligned} \tag{13}$$

where $q_\phi^*(z_{t-1})$ denotes the marginal distribution of z_{t-1} in the variational approximation to the posterior $q_\phi(z_{1:t-1}|\tilde{d}_{1:T}, x_{1:T}, z_0)$. Specifically, $q_\phi^*(z_{t-1})$ is as follows,

$$q_\phi^*(z_{t-1}) = \mathbb{E}_{q_\phi^* z_{t-2}}[q_\phi(z_{t-1}|z_{t-2}, \tilde{d}_{t-1:T}, x_{t-1:T})], \tag{14}$$

and $\tilde{d}_t$ is an exact approximation of d_t with a delta-function. The first term in Eq. (13) is an expected log likelihood under $\mathbb{E}_{q_\phi^*(z_{t-1})}$, which is the reconstruction loss between the input adjacency matrix and the reconstructed adjacency matrix. The second term is the KL-loss, which is the Kullback-Leibler divergence between two distributions.

By applying the dynamic variational graph autoencoder network, we obtain uncertainty embeddings for each node at different timestamp t through Eq. (10). Since unknown class instances are assumed to not occur in the training and validation stage[1], we choose the $k\%$ nodes with the largest entropy as the "expected unknown class nodes" (depicted as S_{unk}), to better learn an accurate classifier for both known and unknown nodes of the testing data. The proposed model consists of a cooperative module, an open-world classification loss and the DVAE loss. They work together to identify whether a node belongs to an existing class or an unknown class. The overall objective function is as follows:

$$\mathscr{L}_{total} = \lambda_1 \mathscr{L}_{OWC} + \lambda_2 \mathscr{L}_{DVAE} \tag{15}$$

where λ_1, λ_2 are balance parameters. The $\mathscr{L}_{OWC}$ and $\mathscr{L}_{DVAE}$ terms represent the open-world classification loss and the dynamic variational autoencoder loss, respectively.

More precisely, the open-world classification loss is used to encourage larger prediction probabilities for the true class of seen class nodes, and to balance the classification output of "expected unknown class nodes" thus avoiding false positive predictions. The open-world classification loss is defined as:

$$\mathscr{L}_{OWC} = \sum_i \mathscr{L}_{OWC}^i \tag{16}$$

and

$$\mathscr{L}_{OWC}^i = \begin{cases} -\log(\hat{y}_{ic}), \ if \ x_i \ is \ from \ class \ c \ \& \ x_i \notin S_{unk} \\ \dfrac{1}{C}\displaystyle\sum_{c=1}^{C} \hat{y}_{ic} log(\hat{y}_{ic}), \ if \ x_i \in S_{unk}. \end{cases} \tag{17}$$

[1]Our settings are relatively stricter than many other open-world learning frameworks, including [22], [36], which used real unseen class samples for training and validation in a semi-supervised way.

where C denotes the number of known classes, $\hat{y}_{i,c}$ is the classification prediction score for the i-th labeled node v_i in the c class through a softmax layer.

$\mathscr{L}_{OWC}$, and $\mathscr{L}_{DVAE}$ are jointly optimized via the proposed objective function, presented in Eq. (15), and all parameters are optimized using the standard backpropagation algorithms.

D. Classification and rejection process

We next introduce how open-world classification and rejection is done during testing. This process is illustrated in Fig. 4. After performing temporal and structure learning and node representation generation, we obtain M different versions of feature vector $(z_i^1, \ldots, z_i^M)$ for each node v_i. To obtain the probability that the node belongs to any unknown class, we utilize the *openmax* [37] method which provides pseudo probability estimation that a node belongs to unknown class space by aggregating calibrated scores from known classes [38]. If the node is identified as "unknown-class sample" by *openmax*, it will be marked as an unknown-class sample. Otherwise, we feed its M different representations into *softmax* to turn them into probabilities over C classes respectively, and we obtain a prediction matrix $S_i \in \mathbb{R}^{M \times C}$. In S_i, each column denotes M different probabilities of a specific class. We choose the largest average column probability and give the related class label as prediction. In summary, the predicted label of a given test sample is predicted through

$$\hat{y}_i = \begin{cases} unknown,\ if\ argmax_{c \in C \cup \{-1\}} P_{open}(c|x_i) = -1 \\ \arg\max_{c \in C} P_{soft}(c|x_i), \quad otherwise. \end{cases} \tag{18}$$

where $P_{open}(c|x_i)$ is obtained from *openmax* and $\arg\max_c P_{open}(c|x_i) = -1$ denotes the average pseudo probability that the unknown class is the largest among all the $C+1$ probabilities and the node is rejected as a sample not from any known class. $P_{soft}(c|x_i)$ is obtained from softmax, and the node's predicted label is the one with the highest average probability among the C known classes.

Therefore, by using a sampling process to generate multiple versions of feature vectors, we are able to assess the confidence that a node belongs to known classes, and automatically obtain a pseudo probability for rejecting nodes that do not belong to any known class.

IV. Algorithm

Algorithm 1 shows the details of the solution. Given an input structured sequence $\mathbb{G}$, the output is the OSSC model, the node representation, and the labels of nodes. First, a dynamic variational graph autoencoder network is built to learn the representation of each node with temporal and structural information encoded (Lines 2-11). The network is built though the DVAE loss and the open-world classification loss (Lines 15-16). By adding up these losses, both deterministic states and stochastic states are trained to capture the uncertainty representations and differentiate known nodes and unknown class nodes.

The complexity of the first layer (GCN) is $O(|E|_{max}DT)$, where D is the dimension of the node features, T is the length of the sequence of the graph sequence, and $|E|_{max}$ is the maximum number of edges that appeared at the same time in the graph. The complexity of the second, third and last layers is $O(h_{max}^2 NT)$, where h_{max} is the maximum length of the embedding generated by these layers, and N is the number of nodes. The total complexity of the proposed method is $O(|E|_{max}DT + h_{max}^2 NT)$ which is linear with respect to the number of graph edges and the number of nodes.

Algorithm 1 The OSSC algorithm

Require: : $\mathbb{G} = \{V, X_t, A_t, Y\}, t = 1, \ldots, T$: structured sequence data with T graph snapshots. $V = V_{train} \cup V_{test}$, $V_{test} = S \cup U$: S are the known classes appearing in X_{train} and U are the unknown classes;
C: the number of known classes.

Ensure: $f(X_{test}) \rightarrow Y, Y \in \{1, \ldots, C, unknown\}$.

1: **while** has not converged **do**
2: // Encoder Model
3: For the first layer:
4: $H_t^{(1)} \leftarrow ReLU(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}X_t W^{(1)})$
5: For the second layer:
6: $H_t^{(2)} = \tanh(\overrightarrow{W}^{(2)}H_t^{(1)} + \overrightarrow{U}H_{t-1}^{(2)} + \overrightarrow{b})$
7: For the third layer:
8: $H_t^{(3)} = \tanh([H_t^{(2)}, H_t^{(1)}]\overleftarrow{W}^{(3)} + H_{t+1}^{(3)}\overleftarrow{U} + \overleftarrow{b})$
9: For the fourth layer:
10: $[\mu_t, \sigma_t] = \tanh(H_t^{(3)}\overrightarrow{W}^{(4)} + H_{t-1}^{(4)}\overrightarrow{U}^{(4)} + \overrightarrow{b}^{(4)})$
11: $Z_T \leftarrow \mu_T + \sigma_T \cdot \zeta, \zeta \sim \mathbb{G}(\mathbf{0}, \mathbf{1})$
12: // Decoder Model
13: $p(A_T^{ij} = 1|z_T^i, z_T^j) \leftarrow \sigma((z_T^i)^T z_T^j)$
14: // Compute Loss
15: $\mathscr{L}_{DVAE} \leftarrow$ obtain the ELBO loss using Eq. (13)
16: $\mathscr{L}_{OWC} \leftarrow$ obtain the open-world classification loss using Eq. (17)
17: Back-propagate loss gradient using Eq. (15)
18: Update the model's weights
19: **if** early stopping condition is satisfied **then**
20: Terminate
21: **end if**
22: **end while**

V. Experiments

We conducted experiments to validate the performance of OSSC. The experiments were run on a workstation equipped with an Intel(R) Xeon(R) Gold 6226R CPU and Nvidia V100S GPU. All the algorithms were implemented using Pytorch and trained with the Adam optimizer. Hyperparameters are tuned by using validation sets. Testing results are reported at the best validation epoch. Experiments include *open classification comparison*, *ablation analysis*, *parameter analysis*, and *efficiency analysis*. To allow reproducibility and future comparisons, codes of the experiments is made available online [2].

[2]https://github.com/sunnyqiny/OSSC

TABLE I: Statistics of the structured sequence datasets

Datasets	# Nodes	# Edges	# Attributes	# Timestamps	# Classes
DBLP-5	6606	42815	100	10	5
Reddit	8291	264050	20	10	4
DBLP-3	4257	23540	100	10	3
Brain	5000	1955488	20	12	10

A. Experimental Setup

Datasets: We use four real-world datasets, i.e., *DBLP-5*[3], *Reddit*[4], *DBLP-3* and *Brain*[5]. *DBLP-5* and *DBLP-3* are co-author network datasets extracted from the DBLP bibliography website. Nodes in *Reddit* represent posts and two nodes are connected if the corresponding posts contain similar keywords. *Brain* is generated from functional magnetic resonance imaging (fMRI) data, where nodes represent tidy cubes of brain tissue and edges indicate the connectivity. Those are typical structured sequence data. A summary of these data is shown in Table I.

Test settings: For each dataset, we hold out part of the class labels as the unknown classes for testing and took the remaining classes as the already seen classes. We randomly sampled 80% of the seen class nodes for training, 10% for validation and 10% for testing. Nodes of unknown classes only appear in the testing set. We varied the number of unknown classes to assess the performance.

Evaluation Metrics: The *Accuracy*, *AUC* and *Macro F1* score were used for evaluation [39].

Implementation Details: The first GCN layer generates node embeddings with a dimension of 256, the second LSTM layer generates deterministic node embeddings with a dimension of 64. Then, the concatenation of the outputs of the first and second layers are given to the third layer to obtain stochastic embeddings of nodes. In the last step, we obtain μ_t and σ_t at the fourth layer through two different linear layers to generate μ_p and σ_p of dimension 16 respectively. We use a fixed learning rate of 5×10^{-3} and a dropout rate of 0.2. The balance parameters λ_1, λ_2 are chosen from a grid search of $\{0.01,1,100\}$.

B. Baselines

The following methods are used as baselines:

- GCN [32]: GCN is a deep convolutional network for graph-structured data, which was originally designed for static networks. First, we apply GCN to each graph snapshot to generate the node representation at each timestamp. Then, we concatenate these representations into a vector and a linear regression is applied to predict the node label. Note that GCN does not have the capability of rejecting the unknown class.
- GAT [40]: GAT leverages masked self-attentional layers to implicitly specify different weights for different neighbor-nodes. We follow the inductive learning framework of GAT. The last graph snapshot is used for testing while the second last graph is utilized for validation and the remaining graphs are used for training. Notice that we removed the nodes from unknown classes from the training sets so that they do not influence the training process. GAT also does not have the capability of rejecting the unknown class.
- GCN$_{thre}$: Based on GCN, we use a probability threshold of $t_i = 0.5$ for classification of each class i to identify the unknown class nodes. It means that if all predicted probabilities are less than the threshold 0.5 for a node, then it will be rejected as an unknown class node. Otherwise, its predicted class is the one that has the highest probability.
- GAT$_{thre}$: Similar with GCN$_{thre}$, we applied GAT to the structured sequences with a threshold of 0.5 to identify unknown nodes.
- OpenWGL [22]: OpenWGL is an open-world graph learning algorithm for static graphs. We applied it to each graph snapshot and the final label of each node was obtained through voting. Note that OpenWGL can identify the unknown class using an automatically determined threshold and it uses real unseen class nodes during training (in the computing of the class uncertainty loss).
- STAR [6]: STAR is a spatio-temporal attentive recurrent network model, which extracts neighborhood representations by sampling and aggregating local neighbor nodes to jointly learn the spatio-temporal contextual information. STAR is an efficient model for dynamic graph node classification. However, it is unable to reject the unknown class.
- STAR$_{thre}$: Same with STAR, the only difference is that a threshold of 0.5 is adopted in the last step to identify nodes of unknown classes.

We compared the proposed method with the above baseline methods, which can be classified into four categories: static graph methods without rejection capability (GCN, GAT), dynamic graph methods without rejection capability (STAR), static graph methods with rejection capability (GCN$_{thre}$, GAT$_{thre}$, OpenWGL) and dynamic graph methods with rejection capability (STAR$_{thre}$).

The baseline algorithms are implemented using Tensorflow (OpenWGL, STAR) and Pytorch (GCN,GAT) trained with the Adam optimizer. We follow the evaluation approach used for open-world learning [41]–[43]. We evaluated all the methods according to the instructions reported in the original papers with the same set of parameter configurations unless otherwise specified, and reported the best results. Except for OpenWGL which use some real unseen class samples during training, the other baselines use the same split in train-validation-test datasets as ours with the setting that no real unseen class samples are used for training and validation.

C. Open Classification Comparison

Table II and Table III list the Accuracy, Macro AUC and Macro F1 score of the methods on an open-world graph node classification task. Table II lists the results on the four datasets with one unknown class (the last class), while Table III lists

[3]https://dblp.uni-trier.de
[4]https://www.reddit.com
[5]https://tinyurl.com/y4hhw8ro

the results on three datasets with two unknown classes [6] (the last two classes). From the results, we make the following observations:

- GCN obtains the worst results since it does not have the capability of rejecting the unknown class. Thus, it cannot handle the temporal evolution in the dynamic structured sequence data. Therefore, all the unknown nodes will be misclassified and the results become worse as the number of unknown nodes increases.
- GAT performs better than GCN since the inductive learning framework can learn representative information from a bunch of graphs simultaneously. However, GAT cannot capture the temporal information, and cannot identify the unknown class properly.
- STAR performs better than both GCN and GAT, which shows the power of adding temporal information into structured sequence learning. It can classify the samples more exactly and result in better performance. However, this method cannot reject the unknown class samples neither.
- GCN$_{thre}$, GAT$_{thre}$ and STAR$_{thre}$ obtain better results than GCN, GAT and STAR respectively, which proves that a threshold can improve the results of detecting unknown nodes. In addition, as the number of unknown nodes increases, GCN$_{thre}$, GAT$_{thre}$ and STAR$_{thre}$ become more competitive.
- OpenWGL obtains better results than GCN$_{thre}$ and GAT$_{thre}$ on all the datasets. It also performs better than STAR$_{thre}$ on most datasets with one or two unknown classes, even if it cannot handle dynamic information. This demonstrates the power of the flexible thresholds, which improves the identification of unknown class samples significantly, compared to the fixed thresholds.
- STAR$_{thre}$ performs better than all the other baselines on Brain, including OpenWGL. We conjecture that Brain is more complex than the other datasets, and STAR achieves better classification result on it, as shown in Table IV.
- The proposed OSSC model consistently outperforms the baselines on the four datasets. This shows that the proposed method can better differentiate between a known class and an unknown class, and capture the label uncertainty in the representation of each node by jointly learning the temporal information and the open-world classification loss.

We also report results in a closed-world learning setting, which means that there is no unknown class, in Table IV. The results show that OSSC reaches competitive results with STAR in a closed-world classification setting, showing its effectiveness and generalization.

D. Ablation Analysis

In this part, we compare variants of OSSC to assess three aspects: (1) the impact of the DVAE loss (KL loss and reconstruction loss), (2) the effect of the open-world classification loss, and (3) the effect of adopting "expected unknown class nodes". The following OSSC variants are designed for comparisons:

- OSSC$_{/v}$: A variant of OSSC with the DVAE loss($\mathscr{L}_{DVAE}$) removed.
- OSSC$_{/o}$: A variant of OSSC with the open-world classification loss ($\mathscr{L}_{OWC}$) removed.

Table V reports the ablation study's results on losses. In terms of the DVAE loss (KL loss and reconstruction loss), OSSC performs better than OSSC$_{/v}$, which confirms that the usage of KL loss can better capture the latent representation of each node, and that the reconstruction loss can preserve node information. In terms of open-world classification loss, comparing OSSC with OSSC$_{/o}$, we can observe that the results of node classification on all the datasets are improved

[6]We omit this experiment on DBLP-3 since it has only three classes in total.

TABLE II: Results on four datasets with an unknown class

Methods	DBLP5			Reddit		
	Acc	AUC	F1	Acc	AUC	F1
GCN	0.2579	0.5007	0.0901	0.1721	0.5283	0.1474
GAT	0.5394	0.7531	0.5307	0.5133	0.4814	0.1904
STAR	0.5443	0.7896	0.5070	0.2045	0.5662	0.2067
GCN$_{thre}$	0.2931	0.5039	0.1011	0.2620	0.4994	0.1985
GAT$_{thre}$	0.5431	0.7660	0.5364	**0.5156**	0.4847	0.1864
OpenWGL	0.5771	0.8528	0.5316	0.4577	0.5159	0.2524
STAR$_{thre}$	0.5540	0.7924	0.5244	0.3260	0.5638	0.2929
OSSC	**0.6110**	**0.8742**	**0.6037**	0.4753	**0.5841**	**0.3026**
Methods	DBLP3			Brain		
	Acc	AUC	F1	Acc	AUC	F1
GCN	0.3114	0.5007	0.1610	0.1271	0.5175	0.0899
GAT	0.3356	0.6111	0.2643	0.2153	0.4581	0.0777
STAR	0.4068	0.7067	0.3527	0.5467	0.9249	0.6831
GCN$_{thre}$	0.3136	0.5009	0.1644	0.3806	0.5006	0.0698
GAT$_{thre}$	0.5015	0.6311	0.4654	0.2888	0.4526	0.0693
OpenWGL	0.5174	0.8188	0.5009	0.3799	0.8841	0.1617
STAR$_{thre}$	0.4265	0.7097	0.3988	0.5533	0.9257	**0.6895**
OSSC	**0.6674**	**0.8433**	**0.5827**	**0.5698**	**0.9367**	0.5025

TABLE III: Results on three datasets with two unknown classes

Methods	DBLP5			Reddit			Brain		
	Acc	AUC	F1	Acc	AUC	F1	Acc	AUC	F1
GCN	0.189	0.504	0.096	0.120	0.509	0.128	0.254	0.449	0.335
GAT	0.325	0.727	0.369	0.126	0.499	0.093	0.135	0.749	0.099
STAR	0.318	0.750	0.369	0.137	0.543	0.148	0.480	0.919	0.593
GCN$_{thre}$	0.595	0.508	0.208	0.343	0.506	0.252	0.468	0.502	0.589
GAT$_{thre}$	0.571	0.777	0.548	0.668	0.503	0.323	0.160	0.751	0.102
OpenWGL	0.634	0.790	0.529	0.604	0.564	0.353	0.474	0.874	0.262
STAR$_{thre}$	0.321	0.751	0.372	0.337	0.546	0.290	0.503	**0.921**	**0.614**
OSSC	**0.657**	**0.855**	**0.570**	**0.677**	**0.581**	**0.371**	**0.509**	0.839	0.548

TABLE IV: Results on four datasets in a closed-world setting

Methods	DBLP5			Reddit		
	Acc	AUC	F1	Acc	AUC	F1
GCN	0.3389	0.5702	0.2434	0.2470	0.5058	0.2217
GAT	0.8124	0.9415	0.8130	0.2759	0.4969	0.1758
STAR	0.8030	0.9550	0.8070	**0.5080**	**0.7500**	**0.5110**
OSSC	**0.8275**	**0.9618**	**0.8226**	0.4700	0.6600	0.4200
Methods	DBLP3			Brain		
	Acc	AUC	F1	Acc	AUC	F1
GCN	0.4695	0.4967	0.2188	0.3760	0.6990	0.3542
GAT	0.5869	0.7104	0.4924	0.1160	0.7872	0.0967
STAR	0.8620	0.9710	0.8670	**0.8920**	**0.9920**	**0.9000**
OSSC	**0.9038**	**0.9726**	**0.9005**	0.7160	0.9588	0.7250

TABLE V: OSSC variants on three datasets

DBLP5	$\|U\| = 1$			$\|U\| = 2$		
	Acc	AUC	F1	Acc	AUC	F1
OSSC$_{/v}$	0.5923	0.8721	0.6002	0.6310	0.8502	0.5681
OSSC$_{/o}$	0.6038	0.8685	0.6035	0.6340	0.8515	0.5690
OSSC	**0.6110**	**0.8742**	**0.6037**	**0.6572**	**0.8547**	**0.5701**
Reddit	$\|U\| = 1$			$\|U\| = 2$		
	Acc	AUC	F1	Acc	AUC	F1
OSSC$_{/v}$	0.4647	0.5840	0.2964	0.6356	0.5798	0.3681
OSSC$_{/o}$	0.4672	0.5563	0.2591	0.6277	0.5804	0.3627
OSSC	**0.4753**	**0.5841**	**0.3026**	**0.6768**	**0.5809**	**0.3710**
Brain	$\|U\| = 1$			$\|U\| = 2$		
	Acc	AUC	F1	Acc	AUC	F1
OSSC$_{/v}$	0.5516	0.9251	0.4982	0.4618	0.8282	0.5369
OSSC$_{/o}$	0.5565	0.9323	0.4715	0.5060	0.8317	**0.5588**
OSSC	**0.5698**	**0.9367**	**0.5025**	**0.5088**	**0.8387**	0.5481

TABLE VI: Evaluation of the effect of "expected unknown class nodes" (S_{unk}).

Accuracy	DBLP5	Reddit	Brain
Without unknown-class nodes	0.5801	0.4435	0.5455
With expected unknown-class nodes	**0.6110**	**0.4753**	**0.5698**

when the open-world classification loss is used, indicating its effectiveness for classification and rejection. To sum up, no matter which loss is removed, the average results of the model decreases. This proves the necessity of using these losses.

Table VI shows the study results on the effect of using "expected unknown class nodes" for training. The first row shows the results obtained without using any unknown class nodes. The second row reports results using "expected unknown class nodes", which are the 10% of known-class samples with the largest entropy (i.e. S_{unk}). We can see that the adoption of pseudo unknown-class nodes significantly improves open-world node classification performance.

E. Parameter Analysis

Feature dimension: We varied the feature dimension of nodes from 2^4 to 2^8 with an increasing step of 2^1, and report results on four datasets with an unknown class in Fig. 5. In terms of accuracy, on the DBLP-5, Reddit and DBLP-3 datasets, as the representation dimension is increased from 8 to 32, the results raise to a peak, and then decrease when the dimension reaches 64. The results on the Brain dataset fluctuates at first. Then, they increase rapidly when the dimension reaches 64. The curve reaches a peak when the embedding dimension is 128. This is another evidence of the complexity of the Brain dataset. However, increasing the embedding dimension from 8 to 128 does not results in a significant improvement. The results in terms of AUC and F1 score are similar to that of accuracy. Results show that when the feature dimension is relatively high, OSSC becomes stable.

VI. Conclusion

Most machine learning models for node classification assume that training and testing data are drawn from the same class label space. Hence, these models can perform poorly when new classes are observed. To address this problem, this paper presented a new Open-world Structural Sequence node Classification method (OSSC) for open-world learning from structured sequences. This problem is challenging because testing samples are from unknown classes that were never seen before.

OSSC uses a GCN-based dynamic variational autoencoder model, which utilizes stochastic states and deterministic states to capture the evolution of node attributes and structures. With the learned latent distribution of each node, a sampling process with truncated Gaussian distribution is adopted to generate multiple versions of node representations, to obtain stable node representations. An open-world classification loss is further utilized to ensure that node representations are sensitive to unknown classes. A combination of *openmax* and *softmax* is adopted to recognize nodes from unknown-classes and to classify others to one of the known-classes. In terms of experiments, a *classification comparison*, *ablation analysis*, *parameter analysis* and *efficiency analysis* were conducted. Results validate the effectiveness and the efficiency of the proposed method.

An interesting future work is to learn the generative framework for open-world structured sequence data learning. Another research direction is to enable the model to recognize new class labels such that not only the unknown-class samples can be identified. Last but not least, features and distributions of these new samples can be explored with the new class labels.

Acknowledgment

This research was supported by Guangdong Provincial Natural Science Foundation under grant no. 2022A1515010129 and Shenzhen Research Foundation for Basic Research, China, under Grant no. JCYJ20210324093000002.

References

[1] D. Xu, W. Cheng, D. Luo, Y. Gu, X. Liu, J. Ni, B. Zong, H. Chen, and X. Zhang, "Adaptive neural network for node classification in dynamic networks," in *IEEE International Conference on Data Mining*. IEEE, 2019, pp. 1402–1407.

[2] A. Ferreira, "Building a reference combinatorial model for manets," *IEEE Network*, vol. 18, no. 5, pp. 24–29, 2004.

[3] P. Flocchini, B. Mans, and N. Santoro, "On the exploration of time-varying networks," *Theoretical Computer Science*, vol. 469, pp. 53–68, 2013.

[4] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2011, pp. 346–359.

[5] O. Michail, "An introduction to temporal graphs: An algorithmic perspective," *Internet Mathematics*, vol. 12, no. 4, pp. 239–280, 2016.

[6] D. Xu, W. Cheng, D. Luo, X. Liu, and X. Zhang, "Spatio-temporal attentive rnn for node classification in temporal attributed graphs." in *International Joint Conferences on Artificial Intelligence*, 2019, pp. 3947–3953.

[7] M. K. Esfahani, P. Kilpatrick, and H. Vandierendonck, "Exploiting in-hub temporal locality in spmv-based graph processing," in *International Conference on Parallel Processing*, vol. 21, 2021.

[8] T. Ben-Nun, M. Besta, S. Huber, A. N. Ziogas, D. Peter, and T. Hoefler, "A modular benchmarking infrastructure for high-performance and reproducible deep learning," in *IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2019, pp. 66–77.

[9] V. Gadiyaram, A. Dighe, S. Ghosh, and S. Vishveshwara, "Network rewiring during allostery and protein-protein interactions: A graph spectral approach," *Allostery*, pp. 89–112, 2021.

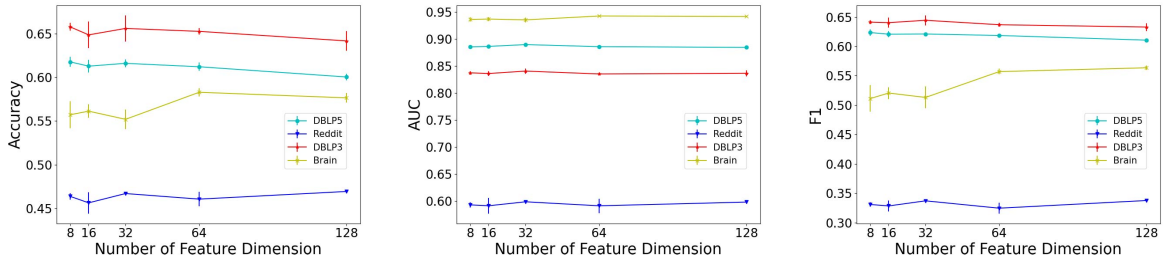

Fig. 5: Node representation under different feature dimensions in terms of Accuracy, AUC and F1 score on four datasets.

[10] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1234–1241.

[11] H. Chereda, A. Bleckmann, F. Kramer, A. Leha, and T. Beissbarth, "Utilizing molecular network information via graph convolutional neural networks to predict metastatic event in breast cancer." in *GMDS*, 2019, pp. 181–186.

[12] L. Dhulipala, G. E. Blelloch, and J. Shun, "Low-latency graph streaming using compressed purely-functional trees," in *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019, pp. 918–934.

[13] X. Sun, Z. Yang, C. Zhang, K.-V. Ling, and G. Peng, "Conditional gaussian distribution learning for open set recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 480–13 489.

[14] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *ACM International Conference on Web Search and Data Mining*, 2021, pp. 833–841.

[15] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multi-scale graph convolution for semi-supervised node classification," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 841–851.

[16] Z. Liu, Y. Fang, C. Liu, and S. C. Hoi, "Relative and absolute location embedding for few-shot node classification on graph," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4267–4275.

[17] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," *arXiv preprint arXiv:1905.10947*, 2019.

[18] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5363–5370.

[19] L. Li, K. Zhao, J. Gan, S. Cai, T. Liu, H. Mu, and R. Sun, "Robust adaptive semi-supervised classification method based on dynamic graph and self-paced learning," *Information Processing and Management*, vol. 58, no. 1, p. 102433, 2021.

[20] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *International Conference on Web Search and Data Mining*, 2020, pp. 519–527.

[21] Y. Ma, Z. Guo, Z. Ren, J. Tang, and D. Yin, "Streaming graph neural networks," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 719–728.

[22] M. Wu, S. Pan, and X. Zhu, "Openwgl: Open-world graph learning," in *IEEE International Conference on Data Mining*. IEEE, 2020, pp. 681–690.

[23] K. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5830–5840.

[24] J. Parmar, S. S. Chouhan, and S. S. Rathore, "Open-world machine learning: Applications, challenges, and opportunities," *arXiv preprint arXiv:2105.13448*, 2021.

[25] J. Hwang, S. W. Oh, J.-Y. Lee, and B. Han, "Exemplar-based open-set

[26] Z. Fang, J. Lu, A. Liu, F. Liu, and G. Zhang, "Learning bounds for panoptic segmentation network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1175–1184.
open-set learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 3122–3132.

[27] P. Perera, V. I. Morariu, R. Jain, V. Manjunatha, C. Wigington, V. Ordonez, and V. M. Patel, "Generative-discriminative feature representations for open-set recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 814–11 823.

[28] Y. Zheng, G. Chen, and M. Huang, "Out-of-domain detection for natural language understanding in dialog systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1198–1209, 2020.

[29] G. Yan, L. Fan, Q. Li, H. Liu, X. Zhang, X.-M. Wu, and A. Y. Lam, "Unknown intent detection using gaussian mixture model with an application to zero-shot intent classification," in *Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 1050–1060.

[30] M. Wu, S. Pan, and X. Zhu, "Openwgl: open-world graph learning for unseen class node classification," *Knowledge and Information Systems*, pp. 1–26, 2021.

[31] L. Galke, B. Franke, T. Zielke, and A. Scherp, "Lifelong learning of graph neural networks for open-world node classification," in *International Joint Conference on Neural Networks*. IEEE, 2021, pp. 1–8.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[33] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," *arXiv preprint arXiv:1605.07571*, 2016.

[34] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "Copod: copula-based outlier detection," in *IEEE International Conference on Data Mining*. IEEE, 2020, pp. 1118–1123.

[35] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda, "Dynamical variational autoencoders: A comprehensive review," *arXiv preprint arXiv:2008.12595*, 2020.

[36] A. R. Dhamija, M. Günther, and T. Boult, "Reducing network agnostophobia," *Neural Information Processing Systems*, vol. 31, 2018.

[37] A. Bendale and T. E. Boult, "Towards open set deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1563–1572.

[38] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, "Generative openmax for multi-class open set classification," *arXiv preprint arXiv:1707.07418*, 2017.

[39] L. Shu, H. Xu, and B. Liu, "Doc: Deep open classification of text documents," *arXiv preprint arXiv:1709.08716*, 2017.

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[41] Y. Tang, Z. Zhao, C. Li, and X. Ye, "Open set recognition algorithm based on conditional gaussian encoder," *Mathematical Biosciences and Engineering*, vol. 18, no. 5, pp. 6620–6637, 2021.

[42] K. Cao, M. Brbic, and J. Leskovec, "Open-world semi-supervised learning," *arXiv preprint arXiv:2102.03526*, 2021.

[43] Z. Wang, B. Salehi, A. Gritsenko, K. Chowdhury, S. Ioannidis, and J. Dy, "Open-world class discovery with kernel networks," in *IEEE International Conference on Data Mining*. IEEE, 2020, pp. 631–640.