Fine-Grained Graph Representation Learning for Heterogeneous Mobile Networks with Attentive Fusion and Contrastive Learning

Shengheng Liu[1, 2 *], **Tianqi Zhang**[1], **Ningning Fu**[1], **and Yongming Huang**[1, 2 *]

[1]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China
[2]Purple Mountain Laboratories, Nanjing, China
{s.liu; huangym}@seu.edu.cn

Abstract

AI becomes increasingly vital for telecom industry, as the burgeoning complexity of upcoming mobile communication networks places immense pressure on network operators. While there is a growing consensus that intelligent network self-driving holds the key, it heavily relies on expert experience and knowledge extracted from network data. In an effort to facilitate convenient analytics and utilization of wireless big data, we introduce the concept of knowledge graphs into the field of mobile networks, giving rise to what we term as wireless data knowledge graphs (WDKGs). However, the heterogeneous and dynamic nature of communication networks renders manual WDKG construction both prohibitively costly and error-prone, presenting a fundamental challenge. In this context, we propose an unsupervised **d**ata-and-**m**odel driven **g**raph **s**tructure **l**earning (DMGSL) framework, aimed at automating WDKG refinement and updating. Tackling WDKG heterogeneity involves stratifying the network into homogeneous layers and refining it at a finer granularity. Furthermore, to capture WDKG dynamics effectively, we segment the network into static snapshots based on the coherence time and harness the power of recurrent neural networks to incorporate historical information. Extensive experiments conducted on the established WDKG demonstrate the superiority of the DMGSL over the baselines, particularly in terms of node classification accuracy.

Code — https://github.com/sh-liu/DMGSL.git
Datasets — https://github.com/sh-liu/WDKG.git

Introduction

The emerging trend of convergence between AI and wireless technology is expected to bring along new research opportunities and better connectivity for people. Now commercial usage of 5G has reached maturity in the leading markets and has sparked a growing appetite for new services that imply extremely stringent requirements. The rapid evolution of networks' capabilities has introduced significant structural complexity, posing challenges for network management and maintenance. As such, the spotlight is on network automation as a prominent trend for forthcoming 6G networks projected for launch by 2030 (You, Huang et al. 2023), which integrates essential functions such as self-configuring, self-optimizing, self-protecting and self-healing (Chi et al. 2023). Achieving such a high degree of network automation requires a fusion of knowledge from both physical models and network big data. Fortunately, knowledge graphs (KGs), a powerful tool to integrate knowledge and data, offer a promising solution for network automation.

While efforts have been made to establish wireless knowledge graphs (WDKGs) (Huang et al. 2024), the existing ones are constructed manually, which is a labor-intensive process with no guarantee of accuracy. The dynamic and heterogeneous nature of wireless communication networks further exacerbates these challenges. The primary hurdle lies in the unprecedentedly enormous and ever-expanding scale of wireless networks. The resultant WDKG includes a vast array of fields and relations, the number of which are both on the order of thousands. Moreover, heterogeneity abounds in the various attributes of nodes (e.g., *block error rate* from the physical layer and *average throughput* from the MAC layer) as listed in Appendix B, alongside multiple types of edges (e.g., causal/explicit/implicit relations). Such extensive heterogeneity complicates manual differentiation and often prone to errors. Additionally, the edges of WDKGs keeps evolving with scene variation (e.g., the mobility of transmitter and receiver), which necessitates near real-time tracking, analysis, and updating. Given these complexities, the development of an automated technique for constructing and refining WDKGs becomes imperative.

Graph structure learning (GSL) methods (Chen, Wu, and Zaki 2020; Franceschi et al. 2019) enable automatic topology construction but traditionally rely on labels for supervision, resulting in biased structures due to the neglect of unlabeled nodes or edges, which limits scalability. To address this issue, self-supervision GSL paradigms have emerged, which leverages supervision signals from contrastive (Liu et al. 2022) or generative learning (Fatemi, El Asri, and Kazemi 2021). Nevertheless, these approaches are primarily tailored for static homogeneous graphs such as Cora, CiteSeer and PubMed (Sen et al. 2008), presenting challenges for structure learning of dynamic and heterogeneous WDKGs. Motivated by these challenges, we propose a novel self-supervised GSL paradigm with attentive fusion and contrastive learning. Our contributions are summarized as follows.

(1) We pioneer GSL on WDKGs. Specifically, we design a data preprocessing scheme to segment and slice network data based on edge properties and coherence time to transform dynamic heterogeneous graphs into a series of static homogeneous graphs, which enables high-fidelity GSL at a fine-grained level.

(2) We propose a novel unsupervised data-and-model driven graph structure learning (DMGSL) method. By integrating a model-driven contrastive learning framework with an expert knowledge adjacency matrix, thus mitigating impact of noisy or anomalous data. The hierarchical attention module and temporal attention module are designed for multidimensional attention fusion.

(3) We apply the proposed method to the real WDKG dataset and evaluate its performance on the node classification task. The results verify that our method outperforms several state-of-the-arts in terms of WDKG structure refinement and the classification performance, as measured by accuracy, precision, recall, and F1-score.

Related Work

Graph Structure Learning

Recent years has witnessed a growing interest in learning graph structures for graph neural networks (GNNs) by modeling the adjacency matrix with learnable parameters and optimizing them alongside GNNs for downstream tasks. Approaches to parameterize the adjacency matrix can be broadly categorized into three types. The first type is model-based (Wang et al. 2021; Franceschi et al. 2019), where the discrete nature of graph structures is taken into account by modeling them as probabilistic models such as Bernoulli and stochastic block models. The second type is based on the similarity matrix (Chen, Wu, and Zaki 2020; Yu et al. 2021), where node similarities are evaluated using various metric functions such as cosine similarity and dot product. The third type treats each entry of the adjacency matrix as a directly learnable parameter (Jin et al. 2020). However, these GSL methods heavily hinge on labeled data for supervision, which can yield biased structures as the learning process prioritizes labeled nodes and edges.

Self-supervision Learning

To extend the applicability of GSL to semi-supervised and unsupervised contexts, self-supervision has emerged. Self-supervision falls into two main categories: generative methods and contrastive methods. The former concentrates on minimizing the reconstruction error, typically achieved through autoencoder (Zhu, Jiao, and Tse 2020; Huang et al. 2022), which aim to preserve essential information of the original data at a pixel-level. Contrastive methods, taking a different approach, aim to train models capable of effectively distinguishing different inputs in the feature space. For instance, Liu et al. (2022) employ self-supervision via mutiview graph contrastive learning, where the mutual information between the anchor view and the learned view is maximized. In comparison to reconstructing the original data, the latter approach is more tractable and scalable.

Dynamic Heterogeneous Graphs Learning

In general, the dynamic graphs can be modeled as snapshot sequences and timestamp graphs. Since the timestamp model can be transformed into a snapshot model with an appropriate granularity, research methods will be the focus of our discussion. Recently there have been a multitude of researches on learning representations of dynamic heterogeneous graphs. There are two main types of approaches. The first type is the incremental method, which leverages the embedding of the last snapshot to learn the current embedding (Wang et al. 2022). This method is computationally efficient but suffers from error accumulation and can only capture short-term temporal information. The second type is the retrained method, which learns embeddings for each snapshot and designs neural networks to capture temporal information (Yang et al. 2020). This approach can capture long-term temporal information but becomes computationally intricate as the number of timesteps increases. To address the computational challenges, self-attention has emerged to selectively learn the most relevant historical information and disregard unnecessary information (Sankar et al. 2020). These methods have been verified efficient in the embedding learning for dynamic heterogeneous graphs, but few researches have focused on the task of structure learning in this context.

Problem Definition

In this paper, the snapshots method is adopted to fit the concept of coherence time in communication network. The coherence time T_c is introduced for dividing dynamic graphs, during which the channel can be reasonably viewed as time-invariant. As a result, the dynamic heterogeneous WDKG can be viewed as a series of static heterogeneous snapshots, denoted as $\mathbb{G} = \left\{\mathcal{G}^{t\cdot T_c} | t = 1, 2, \cdots, T\right\}$, where T is the number of snapshots. For each static heterogeneous graph $\mathcal{G}^t$, the nodes represent various physical parameters in a wireless communication network which belong to different categories.[1] The edges between these nodes represent the relationships among these physical parameters, which can be causal, explicit, or implicit. So each snapshot is represented as $\mathcal{G}^t = (\mathbf{X}^t, \mathcal{V}, \mathcal{E}^t) = (\mathbf{X}^t, \mathbf{A}^t)$ (T_c is omitted for convenience, so as in the following of this paper) where $\mathcal{V}$ is a shared node set and $n = |\mathcal{V}|$ represents the number of nodes, $\mathcal{E}^t$ represents edge set and $m = |\mathcal{E}^t|$ is the number of edges at time step t, $\mathbf{X}^t \in \mathbb{R}^{n\times d}$ is the node feature matrix at time t (the i-th row x_i^t is the feature vector of node v_i at time step t), $\mathbf{A}^t \in [0, 1]^{n\times n}$ is the adjacency matrix (where a_{ij} is the weight of the edge from v_i to v_j at time step t). Considering the heterogeneity of each snapshot, $\mathbf{A}^t = \{\mathbf{A}_1^t, ..., \mathbf{A}_s^t\}$, where s is the number of edge categories. Given a dynamic heterogeneous graph $\mathbb{G}$, our target is to refine the graph structure based on the existing graph structure and feature matrix.

Methodology

In this section, the proposed framework of DMGSL will be explained in detail, the framework is depicted in Fig. 1.

[1]Please find Appendix B for more details.

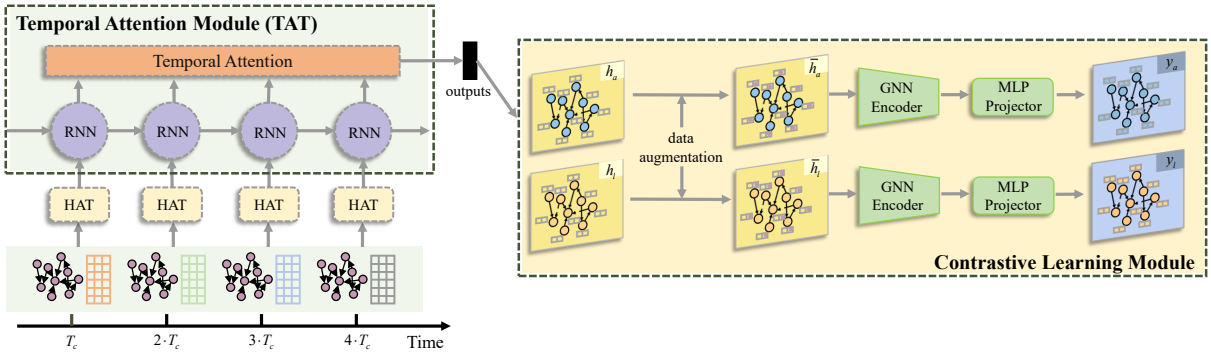


Figure 1: Overall architecture of the proposed DMGSL. It consists of three modules: a) Hierarchical attention module (HAT). The input is $\mathbf{A}^t$ and $\mathbf{X}^t$; b) Temporal attention module (TAT). The input is the anchor graph and learned graph integrating different types of edge, the output is the anchor graph and learned graph integrating temporal information; c) Contrastive learning module. Calculate the contrastive loss, providing a self-supervised signal for unsupervised GSL.

Hierarchical Attention Module

Given the heterogeneous nature of the wireless network, it is sliced in accordance with various relations (or edges). In terms of the dataset we use in this paper, three distinct sliced sub-networks are acquired. Then a hierarchical attention module is devised to independently learn each slice and subsequently merge them using an attention mechanism, which facilitates a nuanced understanding of the discrepancy in relations, allowing for fine-grained learning. The diagram of hierarchical attention module (HAT) is shown in Fig. 2.

Anchor Graphs and Learned Graphs. As defined before, the dynamic heterogeneous WDKG is denoted as a series of static snapshots $\mathbb{G} = \{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^T\}$. As to the static graph at t-th snapshot $\mathcal{G}^t$, the adjacency matrix $\mathbf{A}^t$ is divided into three sub-adjacency matrices according to different kinds of relations (or edges), i.e., $\mathbf{A}_1^t$, $\mathbf{A}_2^t$, $\mathbf{A}_3^t$. Combined with $\mathbf{X}^t$ (the feature matrix at time t), we get $\mathbf{E}_{1,a}^t = (\mathbf{X}^t, \mathbf{A}_1^t)$, $\mathbf{E}_{2,a}^t = (\mathbf{X}^t, \mathbf{A}_2^t)$, $\mathbf{E}_{3,a}^t = (\mathbf{X}^t, \mathbf{A}_3^t)$ as the initial matrices of anchor graphs.

To acquire the initial matrices of learned graphs, a full graph parameterization (FGP) learner is considered to generate sketchy adjacency matrix of WDKG from feature matrix at time t. The FGP learner parameterizes each element of the adjacency matrix independently, the learned adjacency matrix can be presented as $\mathbf{A}_s^t$. Then $\mathbf{E}_{1,l}^t = (\mathbf{X}^t, \mathbf{A}_s^t)$, $\mathbf{E}_{2,l}^t = (\mathbf{X}^t, \mathbf{A}_s^t)$, $\mathbf{E}_{3,l}^t = (\mathbf{X}^t, \mathbf{A}_s^t)$ are obtained as the initial matrices of learned graphs.

Hierarchical Attention Model. The categories of relations between fields are different, including causal, explicit and implicit relations. Therefore, we slice the wireless network into three sub-networks and learn the information of them separately. The common method is to average the learned information from three slices, but in fact these slices are of different importance to structure learning. For instance, the direct influence of the causal relation (or edge) on the structure is higher than the indirect influence of the implicit relation (or edge). Therefore, we introduce a hierarchical attention model to learn the importance of different edges to GSL of wireless network, so as to integrate the information of the three network slices more explainably.

Specifically, the initial matrices of anchor graphs and learned graphs (i.e. $\mathbf{E}_{1,a}^t$, $\mathbf{E}_{2,a}^t$, $\mathbf{E}_{2,a}^t$, $\mathbf{E}_{1,l}^t$, $\mathbf{E}_{2,l}^t$, $\mathbf{E}_{3,l}^t$) are firstly entered into a nonlinear transformation function so that they map to the same feature space by $\sigma\left(\mathbf{W} \cdot \mathbf{e}_{s,a}^t + \mathbf{b}\right)$, $\sigma\left(\mathbf{W} \cdot \mathbf{e}_{s,l}^t + \mathbf{b}\right)$, where $\mathbf{e}_{s,a}^t, e_{s,l}^t \in \mathbb{R}^d$ are the transpose of i-th row of initial matrices $\mathbf{E}_{s,a}^t, \mathbf{E}_{s,l}^t \in \mathbb{R}^{n\times(d+n)}$ (i is omitted for convenience) which are two matrix representations of WDKG, σ denotes the activation function, $\mathbf{W}$ and $\mathbf{b}$ represent the weight matrix and bias vector, parameters of which are shared by anchor and learned graphs in the same edge-level (e.g., $\mathbf{E}_{1,a}^t$ and $\mathbf{E}_{1,l}^t$ share the parameters of $\mathbf{W}$ and $\mathbf{b}$). The mapped graphs are denoted as $h_{1,a}^t$, $h_{2,a}^t$, $h_{3,a}^t$ and $h_{1,l}^t$, $h_{2,l}^t$, $h_{3,l}^t$. Then, the similarities between the mapped graphs and the edge-level attention parameterized vector are calculated to evaluate the importance of different slices to structure learning. Sequently, the normalized weight factors of the anchor graphs and learned graphs with edge type s at time t are figured up, which are denoted as $\alpha_{s,a}^t$, $\alpha_{s,l}^t$. The process can be defined as:

$$\begin{aligned}
\alpha_{s,a}^t &= \frac{\exp\left(\mathbf{q}^\top \cdot \sigma\left(\mathbf{W} \cdot \mathbf{e}_{s,a}^t + \mathbf{b}\right)\right)}{\sum_s \exp\left(\mathbf{q}^\top \cdot \sigma\left(\mathbf{W} \cdot \mathbf{e}_{s,a}^t + \mathbf{b}\right)\right)}, s \in [1, m] \\
\alpha_{s,l}^t &= \frac{\exp\left(\mathbf{q}^\top \cdot \sigma\left(\mathbf{W} \cdot \mathbf{e}_{s,l}^t + \mathbf{b}\right)\right)}{\sum_s \exp\left(\mathbf{q}^\top \cdot \sigma\left(\mathbf{W} \cdot \mathbf{e}_{s,l}^t + \mathbf{b}\right)\right)}, s \in [1, m]
\end{aligned} \tag{1}$$

Lastly, the graphs with single communication relation can be merged with the normalized weight factors. The transpose of i-th row in the merged matrices $\mathbf{E}_a^t, \mathbf{E}_l^t \in \mathbb{R}^{n\times(d+n)}$ are expressed as $\mathbf{e}_a^t$ and $\mathbf{e}_l^t$, which can be viewed as two kinds of embeddings of node i in the wireless communication network at time t. The merge can be formulized as:

$$\mathbf{e}_a^t = \sum_{s=1}^{m} \alpha_{s,a}^t \cdot \mathbf{e}_{s,a}^t, \quad \mathbf{e}_l^t = \sum_{s=1}^{m} \alpha_{s,l}^t \cdot \mathbf{e}_{s,l}^t. \tag{2}$$

By performing the above operations on the WDKG at each time step, the anchor graphs and the learned

graphs of each snapshot can be obtained, denoted as h_a^t and h_l^t, the corresponding matrices are represented as $\left\{\mathbf{E}_a^1, \mathbf{E}_a^2, \cdots, \mathbf{E}_a^T \in \mathbb{R}^{n\times D}\right\}$, $\left\{\mathbf{E}_l^1, \mathbf{E}_l^2, \cdots, \mathbf{E}_l^T \in \mathbb{R}^{n\times D}\right\}$, where n is the number of nodes and D is the output dimension of hierarchical attention model.

Temporal Attention Module

The wireless communication system encompasses a highly complex channel, including a series of channels caused by obstacles such as reflection, diffraction, scattering, coherence and shadowing. The mobility of transmitter or receiver renders time-invariance elusive. To address this, the notion of coherence time is introduced, representing the duration in which the channel can reasonably be viewed as time-invariant. Assuming the transmitter is fixed, the coherence time can be calculated based on the radio frequency and the radial velocity of receiver. Consequently, the dynamic WDKG is partitioned into several static snapshots accordingly. On this basis, the temporal attention module (TAT) is introduced to effectively integrate the abundant temporal features inherent in the WDKG.

Long Short-term Memory Unit. In this paper, to model the dynamic information of WDKG, we employ a basic variant of RNN called long short-term memory (LSTM) which is qualified for conveying information during a long time. To cater to the limited memory units, LSTM preserves useful information that needs long-term memory, and forgets superfluous information. Moreover, a mechanism that can dynamically adjust memory is introduced to update the valuable information that needs to be remembered in time. Specifically, the LSTM model contains state vector $\mathbf{s}^t$, forgetting vector $\mathbf{f}^t$, memory vector $\mathbf{c}^t$, input vector $\mathbf{i}^t$, output vector $\mathbf{o}^t$. The matrices output from hierarchical attention model are represented as $\left\{\mathbf{E}_a^1, \mathbf{E}_a^2, \cdots, \mathbf{E}_a^T \in \mathbb{R}^{n\times D}\right\}$, $\left\{\mathbf{E}_l^1, \mathbf{E}_l^2, \cdots, \mathbf{E}_l^T \in \mathbb{R}^{n\times D}\right\}$, the transpose of their i-th row (denoted as $\mathbf{e}_a^t, \mathbf{e}_l^t \in \mathbb{R}^D$, where i is omitted) are entered into LSTM. An LSTM unit can be represented by the following formulas (omit the subscripts of anchor and learned graphs):

$$\begin{aligned}
\mathbf{i}^t &= \sigma\left(\mathbf{W}_i \cdot \left[\mathbf{e}^t \parallel \mathbf{s}^{t-1}\right] + \mathbf{b}_i\right), \\
\mathbf{f}^t &= \sigma\left(\mathbf{W}_f \cdot \left[\mathbf{e}^t \parallel \mathbf{s}^{t-1}\right] + \mathbf{b}_f\right), \\
\mathbf{o}^t &= \sigma\left(\mathbf{W}_o \cdot \left[\mathbf{e}^t \parallel \mathbf{s}^{t-1}\right] + \mathbf{b}_o\right), \\
\tilde{\mathbf{c}}^t &= \tanh\left(\mathbf{W}_c \cdot \left[\mathbf{e}^t \parallel \mathbf{s}^{t-1}\right] + \mathbf{b}_c\right), \\
\mathbf{c}^t &= \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t, \mathbf{s}^t = \mathbf{o}^t \odot \tanh\left(\mathbf{c}^t\right),
\end{aligned} \tag{3}$$

where $t \in \{1, 2, \cdots, T\}$, $\mathbf{i}^t$, $\mathbf{f}^t$, $\mathbf{o}^t$, $\mathbf{c}^t \in \mathbb{R}^F$ (F is the output dimension of LSTM), $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o$, $\mathbf{W}_c \in \mathbb{R}^{F\times 2D}$ and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, $\mathbf{b}_c \in \mathbb{R}^F$ are trainable parameters, σ is the activation function, $\parallel$ is the concatenation operation, $\odot$ is the Hadamard product. $\mathbf{s}^0$, $\mathbf{c}^0 \in \mathbb{R}^D$ need to be initialized, and in this article they are initialized to an all-one vector $\mathbf{1}$. The outputs of LSTM (i.e., the graph structure matrix at t time step that has integrated historical information) can be defined as $\mathbf{S}^t = \left[\left(\mathbf{s}_1^t\right)^\mathsf{T}, \left(\mathbf{s}_2^t\right)^\mathsf{T}, \cdots, \left(\mathbf{s}_n^t\right)^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{n\times F}$.

Temporal Attention Model. To fuse the acquired wireless network topology across different snapshots, we employ a temporal attention model. This model calculates contribution factors to determine the impact of various snapshots on the overall structure learning process. For example, snapshots with lower doppler effect (i.e., a lower radial velocity between transmitter and receiver) may be more significant in the learning of network topology compared to other snapshots. These contribution factors enable us to effectively weigh the influence of each snapshot on the overall structure learning process. The i-th row of the graph structure at all times are extracted to constitute a fresh matrix $\mathbf{S}_i = \left[\left(\mathbf{s}_i^1\right)^\mathsf{T}, \left(\mathbf{s}_i^2\right)^\mathsf{T}, \cdots, \left(\mathbf{s}_i^T\right)^\mathsf{T}\right]^\mathsf{T}, \mathbf{s}_i^1, \mathbf{s}_i^2, \cdots, \mathbf{s}_i^T \in \mathbb{R}^F$, which is the input. The popular scaling dot multiplication attention mechanism in natural language processing is adopted. The input $\mathbf{S}_i \in \mathbb{R}^{T\times F}$ is multiplied with three parameter matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V \in \mathbb{R}^{F\times F'}$, mapping into different feature spaces, represented as $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V} \in \mathbb{R}^{T\times F'}$, which is viewed as the linear transformation of the input. Using $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ rather than $\mathbf{S}_i \in \mathbb{R}^{T\times F}$ in the calculation enhances the fitting ability of the model effectively. Then, multiply $\mathbf{Q}$ and $\mathbf{K}^\mathsf{T}$ to generate the similarity matrix. What needs to be emphasized is that when the dimension of $\mathbf{K}$ increases, the variance of $\mathbf{Q}\cdot\mathbf{K}^\mathsf{T}$ will become larger. In order to reduce the variance, each element of the similarity matrix is divided by $\sqrt{F'}$ (F' is the dimension of $\mathbf{K}$. The normalized similarity matrix can be regarded as a weight matrix. Finally, multiply the weight matrix with $\mathbf{V}$ and calculate the weighted sum, which is the output of the network. The above process can be formulized as follows:

$$\mathbf{Z}_i = \Gamma_i \cdot \mathbf{V}_i = \text{softmax}\left(\frac{\mathbf{Q}\cdot\mathbf{K}^\mathsf{T}}{\sqrt{F'}} + \mathbf{M}\right)\cdot\mathbf{V}, \tag{4}$$

where $\Gamma_i \in \mathbb{R}^{T\times T}$ is the weight matrix, $\mathbf{M} \in \mathbb{R}^{T\times T}$ is the masking matrix. If $M_{u\eta} = -\infty$, there is no effect from time u to η, and the corresponding element in the weight matrix is 0. If u is earlier than η, then $M_{u\eta} = 0$; Otherwise, $M_{u\eta} = -\infty$. The output of the temporal attention model is defined as $\mathbf{Z}_i = \left[\left(\mathbf{z}_i^1\right)^\mathsf{T}, \left(\mathbf{z}_i^2\right)^\mathsf{T}, \cdots, \left(\mathbf{z}_i^T\right)^\mathsf{T}\right]^\mathsf{T}, \mathbf{z}_i^1, \mathbf{z}_i^2, \cdots, \mathbf{z}_i^T \in \mathbb{R}^{F'}$, where F' is the output dimension.

In order to enhance the performance, the extended multi-head attention mechanism is used. To be concrete, we define multiple groups (i.e. κ groups) of $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$, each group is calculated separately to generate different $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, and learn various parameters, the obtained multiple outputs are concatenated to get $\mathbf{Z}_i = \text{Concat}\left(\hat{\mathbf{Z}}_i^1, \hat{\mathbf{Z}}_i^2, \cdots, \hat{\mathbf{Z}}_i^\kappa\right)$. In this paper, we take all the $\mathbf{z}_i^T$ in $\mathbf{Z}_i$ to constituent matrix $\mathbf{E} = \left[\left(\mathbf{z}_1^T\right)^\mathsf{T}, \left(\mathbf{z}_2^T\right)^\mathsf{T}, \cdots, \left(\mathbf{z}_n^T\right)^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{n\times F'}$ as output. The anchor graph and learned graph formed from temporal attention module can be denoted as h_a and h_l which are regard as two graph representations of the wireless network.

Contrastive Learning Module

Data augmentation is significant method to mitigate overfitting and explore richer information. We use two common data augmentation schemes, edge dropping and feature

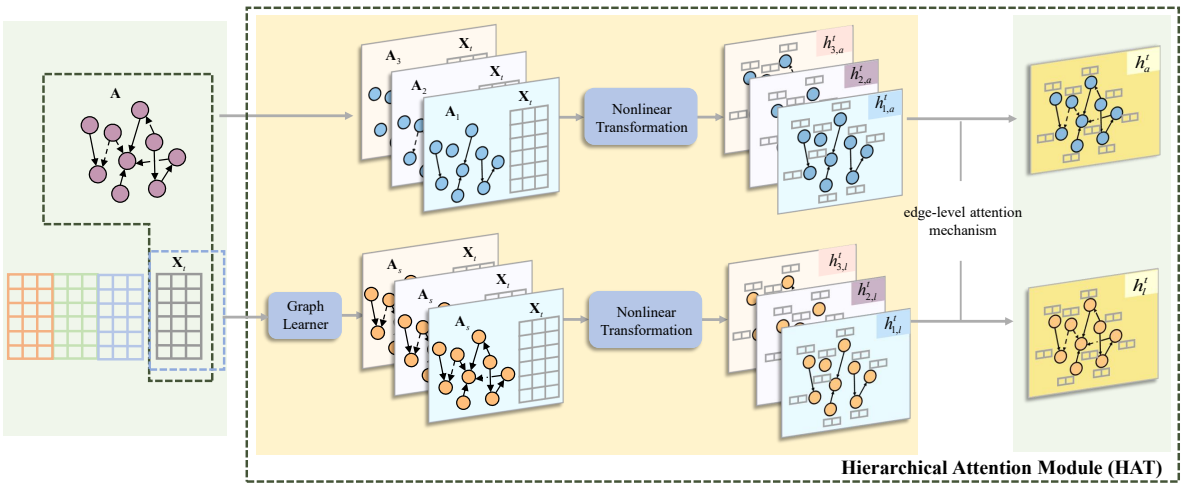


Figure 2: Schematic diagram of hierarchical attention module. The input is $\mathbf{A}^t$ and $\mathbf{X}^t$. In the upper branch, $\mathbf{A}^t$ is divided into three sub-adjacency matrices (causal/implicit/explicit relations) and mapped to a new feature space along with feature matrix. In the lower branch, an adjacency matrix is learned by a graph learner and then mapped to a new feature space along with feature matrix. The three anchor graphs and learned graphs are fused respectively through the edge-level attention mechanism.

masking, the augmented matrices $\bar{\mathbf{E}}_a$ and $\bar{\mathbf{E}}_l$ are obtained, corresponding to the two augmented views $\bar{h}_a$ and $\bar{h}_l$.

Next, the augmented graphs are encoded and compressed, transforming the high-dimension into lower dimension. Graph convolutional network (GCN) is exploited as the encoder, the encoding process can be expressed as

$$\mathbf{H}_a = \mathrm{GCN}_\theta\left(\bar{\mathbf{E}}_a\right), \mathbf{H}_l = \mathrm{GCN}_\theta\left(\bar{\mathbf{E}}_l\right), s \in [1, m], \tag{5}$$

where θ is the parameter of GCN encoder, $\mathbf{H}_a, \mathbf{H}_l \in \mathbb{R}^{n \times d_1}$ (d_1 represents the output dimension of encoder) are the encoded structure matrices.

Furthermore, to calculate the contrast loss function, we reflect the views to another latent space with the assistance of multiple layer projection (MLP), which is formalize as:

$$\mathbf{Y}_a = g_\varphi\left(\mathbf{H}_a\right), \mathbf{Y}_l = g_\varphi\left(\mathbf{H}_l\right) \tag{6}$$

where φ is the parameter of the projector $g_\varphi(\cdot)$, $\mathbf{Y}_a, \mathbf{Y}_l \in \mathbb{R}^{n \times d_2}$ (d_2 is the output dimension of projector) are mapped graph matrices of the anchor and learned graphs. Then, a contrast learning loss function (van den Oord, Li, and Vinyals 2019) is used to maximize the similarity between each row vector of the two graph structures:

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} \left[\ell\left(\mathbf{y}_{a,i}, \mathbf{y}_{l,i}\right) + \ell\left(\mathbf{y}_{l,i}, \mathbf{y}_{a,i}\right)\right], \tag{7}$$

$$\ell\left(\mathbf{y}_{a,i}, \mathbf{y}_{l,i}\right) = \log \frac{e^{\mathrm{sim}\left(\mathbf{y}_{a,i}, \mathbf{y}_{l,i}\right)/p}}{\sum_{k=1}^{n} e^{\mathrm{sim}\left(\mathbf{y}_{a,i}, \mathbf{y}_{l,i}\right)/p}}, \tag{8}$$

where $\mathrm{sim}(\cdot,\cdot)$ is the cosine similarity function, p is the temperature parameter. The loss function is minimized during the training process, where involves iteratively updating the model's parameters using gradient descent. [2]

[2]We employ a bootstrapping mechanism controlled by a hyperparameter τ in each iteration to balance the contribution of expert knowledge and data during each learning iteration. Please find Appendix C for more details.

Experiments

In this section, we conduct a series of experiments to examine the effectiveness of the proposed framework for fine-grained graph representation learning from the following aspects. First, we compare the learned adjacency matrix with expert knowledge and other baselines. Second, we exploit the learned structure to perform node classification tasks and compare the results with baselines. Finally, we investigate the influence of key hyperparameters on the performance of our proposed method, aiming to provide guidelines for parameter tuning. The detailed setup of experiments can be found in Appendix A.

Baseline

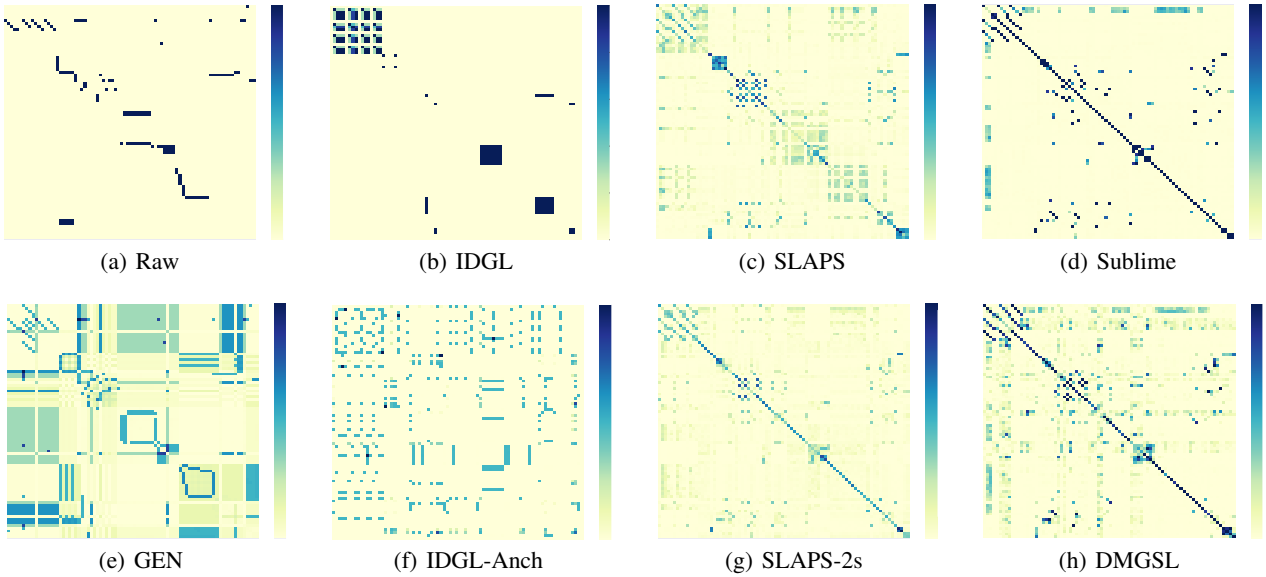
We compare the proposed method with state-of-the-art methods of structure learning, including Sublime (Liu et al. 2022), GEN (Wang et al. 2021), IDGL (Chen, Wu, and Zaki 2020), SLAPS (Fatemi, El Asri, and Kazemi 2021), and their variant, IDGL-Anch and SLAPS-2s.

Dataset

The dataset is collected by an automated guided vehicle which travels in the park at a speed of 10 km/h for 500 m along a planned route, receiving signals with a frequency of 3300 − 3800 Hz, recording 40 data per second. We processed and organized the data, the details of dataset we use are provided in Appendix B.

Comparisons

In the proposed framework, each element of the output adjacency matrix represents the probability of an edge existing between two nodes; the higher the value, the closer the correlation between two nodes. We depict the heatmap of the original adjacency matrix and the adjacency matrices learned from the proposed method and other baselines, as



(a) Raw (b) IDGL (c) SLAPS (d) Sublime

(e) GEN (f) IDGL-Anch (g) SLAPS-2s (h) DMGSL

Figure 3: Heatmaps of adjacency matrices.

Dataset	Method	Accuracy	Precision	Recall	F1-score
Uplink throughput data (35min)	IDGL	0.4375 +/- 0.1046	0.3515 +/- 0.1467	0.4375 +/- 0.1046	0.373 6+/-0.1343
	IDGL-Anch	0.3625 +/-0.1000	0.2221 +/-0.1367	0.3625 +/- 0.1000	0.2571 +/- 0.1332
	SLAPS	0.5875 +/- 0.0637	0.4764 +/- 0.1268	0.5875 +/- 0.0637	0.5100 +/- 0.0974
	SLAPS-2s	0.6000 +/- 0.0637	0.5525 +/- 0.0788	0.6000 +/- 0.0729	0.5224 +/- 0.0766
	GEN	0.5750 +/- 0.0729	0.5439 +/- 0.0723	0.5750 +/- 0.0729	0.5302 +/- 0.0632
	Sublime	0.6125 +/- 0.0468	0.5112 +/- 0.0644	0.6125 +/- 0.0468	0.5438 +/- 0.0550
	DMGSL (w/o TAT)	0.6125 +/- 0.0250	0.5639 +/- 0.0204	0.6125 +/- 0.0250	0.5538 +/- 0.0356
	DMGSL (w/o HAT)	0.6625 +/- 0.0306	0.6108 +/- 0.0417	0.6625 +/- 0.0306	0.6104 +/- 0.0350
	DMGSL	**0.7000 +/- 0.0250**	**0.6373 +/- 0.0321**	**0.7000 +/- 0.0250**	**0.6436 +/- 0.0321**

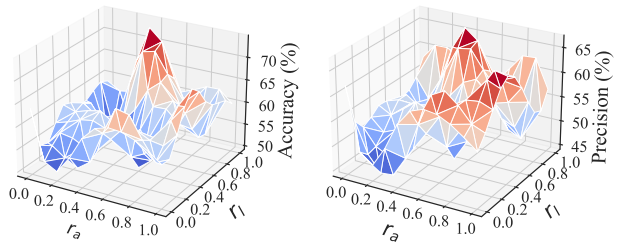
Table 1: Performance of node classification (values with standard deviation). The highest is highlighted with **boldface**, the second highest is highlighted with underline.

shown in Fig. 3. It should be noticed that the darker the patch, the more likely an edge exists between the two nodes. It can be seen that there are only a few obvious deepened color blocks in the raw adjacency matrix, which reflects limited correlations. In contrast, some baselines have learned relations concentrated near the diagonal (IDGL, SLAPS), while others are extremely scattered (IDGL-Anch). Compared to GEN, SLAPS-2s, and Sublime, the adjacency matrix learned by our proposed method presents more relations and avoids learning unnecessary relationships. To a certain extent, it modifies and refines the raw matrix effectively.

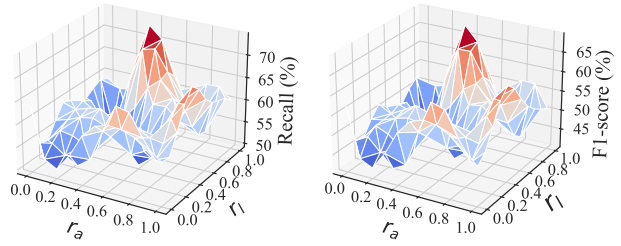
To objectively and quantitatively evaluate the effectiveness of the proposed method, we test a node classification task, whose performance is quantified with accuracy, precision, recall and F1-score. We compare the performance of the proposed DMGSL with several prevalent baselines. From Table 1, we observe that DMGSL outperforms the other baselines and has clear advantages. This can be attributed to its focus on the nature of mobile communication networks. By partitioning the network hierarchically and chronologically and narrowing the learning unit, the utilization of the measured network data is maximized. Other approaches, on the other hand, process all the data together, neglecting the different data fields and time-variance of mobile network, resulting in learned structures that fail to mine detailed endogenous relations.

Impact of Key Hyperparameter

Since the data fields (or nodes) and relations (or edges) of wireless communication data is less than that of general public datasets, it is significant to prevent overfitting in the learning process. To this end, we perform data augmentation by feature masking before encoding and mapping. The feature masking rate of the anchor graph r_a and learned graph r_l are two important hyperparameters, we change these two parameters respectively in several experiments. Fig. 4 shows the classification performance under different feature masking rate. It can be seen that both small and large r_a cor-



(a) Accuracy versus mask rate. (b) Precision versus mask rate.



(c) Recall versus mask rate. (d) F1 score versus mask rate.

Figure 4: Impact of mask rate on the classification performance.

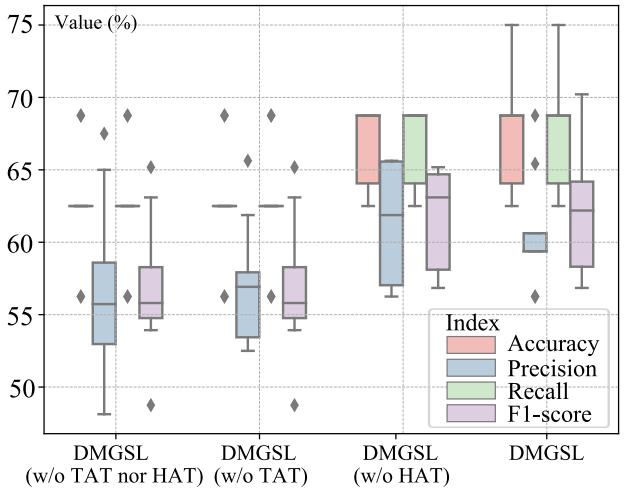


Figure 5: Performance with different model configurations.

respond to poor classification performance, which is because of overfitting and underfitting respectively during contrastive learning. The results are in agreement with our theoretical analysis. The model performs best when $r_l = 0.8$ and $r_a = 0.4$ in point of four metrics. [3]

Ablation Study

To verify the validity of the hierarchical attention model and temporal attention model introduced in our proposed method, we conduct an ablation study. Node classification performance is employed to evaluate if each component positively contributes to the final learned structure, as shown in Table 1 and Fig. 5. Firstly, considering only the hierarchical attention model (DMGSL (w/o TAT)), the classification accuracy is higher than other baselines and comparable to Sublime, but the precision and F1-score is better than Sublime. Then, considering only the temporal attention model (DMGSL (w/o HAT)), the performance is significantly higher than the other baselines. Finally, both attention models are introduced together (DMGSL), it is obvious that the classification performance is further improved. Meanwhile, it can be seen in Fig. 5 that when there is no attention models, the performance is relatively worst compared to other configurations. Therefore, both the hierarchical attention model and temporal attention model enhance classification performance.

So far, we have demonstrated that DMGSL performs well on node classification. But whether it means good performance on structure learning? It is true to some degree. Although the process of node classification seems that only node attributes are paid attention, it is important to note that the impact of edges is also considered while learning the network structure and node embedding. Therefore, the high performance achieved in the node classification task indicates that the learned network structure is meaningful. It not only incorporates the attributes of wireless network nodes but also captures the relations between nodes through edges, indeed supplementing and optimizing expert knowledge. By successfully constructing the network structure, our method provides a potential foundation for network automation.

Due to the difficulty in obtaining a dataset that exhibits both heterogeneity and dynamic features similar to the data acquired from mobile networks, we plan to collect more datasets using our testbed with different configurations and/or conditions to evaluate the generalizability in our subsequent works. Refining our method to extend its applicability to a broader range of datasets will be an important direction of future work.

Conclusion

The avenue of AI is fast evolving and will impact the telecom industry in the years to come. This study selects network automation as a cutting-in point and considers the problem of automatic graph representation for the next-generation mobile networks. Based on edge properties, the presented framework incorporates coherence time to partition dynamic networks into static snapshots. Hierarchical attention independently learns and merges slices, facilitates nuanced understanding of relationships, while the temporal attention integrates temporal features for enhanced learning. Furthermore, the method employs LSTM and multi-head attention mechanisms to capture temporal dynamics. Data augmentation techniques such as edge dropping and feature masking are utilized to mitigate overfitting and extract richer information from data. Overall, this research marks significant progress in autonomous learning and refinement of network topology for wireless communications, which paves the way for advancements in network automation.

[3] Due to space limit, other hyperparameters, such as learning rate, update frequency, and neighborhood size, are discussed more thoroughly in Appendix C.

A Setup of Experiments

The pre-processed network data, i.e., feature matrix, is input into the proposed framework together with the adjacency matrix of expert knowledge for training, and the output is learned adjacency matrix and node embedding. To intuitively reflect the reliability of the learned structure, we use the learned node embedding for the node classification task. The training, validation, and test sets are divided by hierarchical sampling at a ratio of $6:2:2$. To comprehensively evaluate the performance of node classification, we calculate four commonly used metrics: accuracy, precision, recall, and F1-score. The proposed method was implemented using an NVIDIA GeForce RTX 3060 Laptop GPU with 13.8 GB of memory. The implementation was based on PyTorch 1.9.0, utilizing the SGD and Adam optimizers.

B Dataset

The dataset can be summarized as follows, with details shown in Table 2 and Table 3.

- The adjacency matrix of the uplink throughput KG, generated by expert knowledge, includes 82 nodes and 133 relations, with three types of relations [4]
- The measured uplink throughput data (with 82 data fields, including throughput capacity, block error rate, etc.) are collected with a window of 15 minutes and 35 minutes, resulting in $38,250$ and $120,418$ pieces of data.

Data	Nodes	Edges	Features
Adjacency matrix	82 (10 classes)	133 (3 classes)	/
Uplink throughput data (15min)	/	/	38250
Uplink throughput data (35min)	/	/	120418

Table 2: Details of the wireless communication dataset.

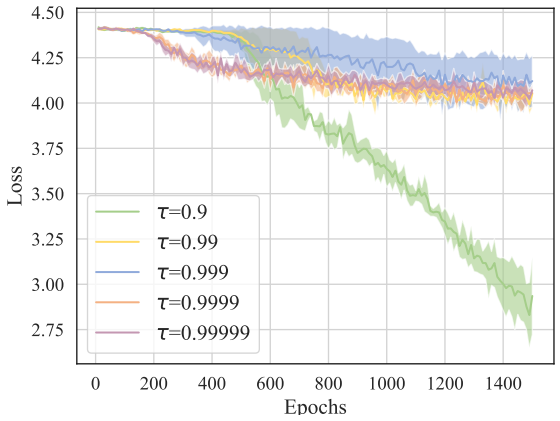
Data field	Protocol layers	Data categories
nr_pdsch_bler	physical layer	block error rate
prb_num_ul_slot	physical layer	frame structure value
nr_phy_throughput_ul	data link layer	flow rate
nr_ul_dl_slot_ratio	network layer	frame structure value

Table 3: Examples of wireless communication data.

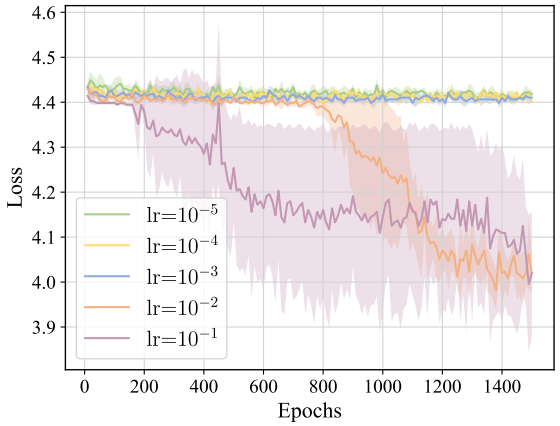
We perform data completion and normalization on the collected network data. The processed data are used as the feature matrix in structure learning.

C Influence of Key Hyperparameters

Our framework is designed to jointly learn the graph structure from both expert-defined adjacency matrices and collected data. To achieve this, we employ a bootstrapping mechanism controlled by a hyperparameter τ, which balances the contribution of expert knowledge and data during each learning iteration. This mechanism updates the anchor graph every 10 epoches. $\tau \in [0,1]$ is a key parameter for adjusting the degree of updating: the closer τ is to 1, the larger the proportion of the anchor graph in update process, meaning the degree of updating is lower. We varied the size of τ and conducted several experiments, plotting the curves of contrastive loss versus training epochs with different τ values, as shown in Fig. 6(a). We readily observe that, when τ is 0.9, the contrastive loss shows a rapid downward trend initially but continues to decrease without a convergence trend as epochs increase. This is due to the rapid variation of the anchor graph, which causes unstable learning. When τ is greater than or equal to 0.99, the curves gradually converge as epochs increase. The curve for $\tau = 0.999$ converges the slowest; despite the curve for $\tau = 0.99$ converges slowly at the beginning, it achieves a stable value faster than $\tau = 0.9999$ and $\tau = 0.99999$. Therefore, we conclude that when $\tau = 0.99$, the model converges to a fair performance and is scalable.



(a) Loss vs. Epoch with various τ.



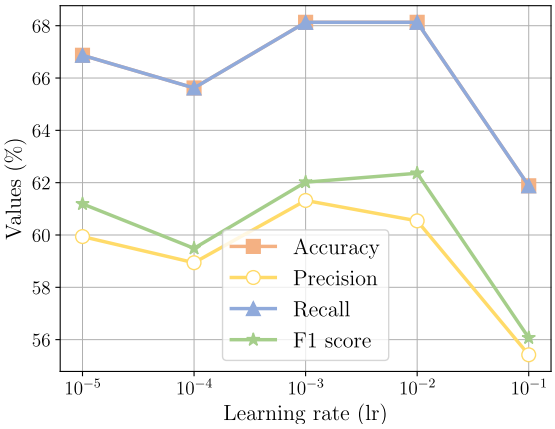
(b) Loss vs. Epoch with various lr.

Figure 6: Training process.

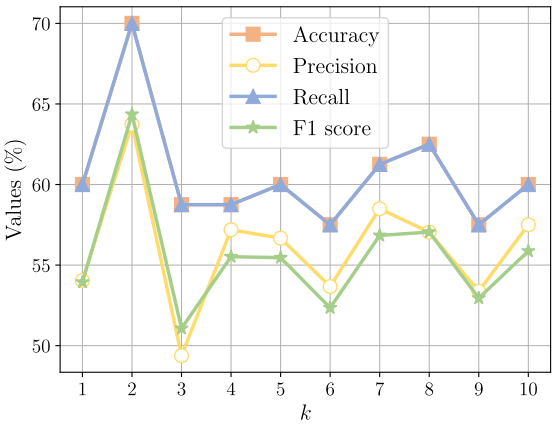
The learning rate, denoted as lr, is another crucial parameter in deep learning, as it affects both the convergence

[4] 1 represents causal relation, 2 represents implicit relation, 3 represents explicit relation.

speed and the effectiveness of learning. In order to find the optimal lr, we set lr to $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$, and evaluated the classification performance. The curves of contrastive loss and classification performance with different lr values are shown in Fig. 6(b) and Fig. 7(a), respectively. From 6(b) we see that, the curves for $\mathrm{lr} = 10^{-5}$, $\mathrm{lr} = 10^{-4}$, $\mathrm{lr} = 10^{-3}$ show only minimal convergence trends as epochs increase. This is attributed to the convergence process slowed down by the small learning rate, which makes it difficult for the model to reach an optimal solution within a reasonable number of iterations. When $\mathrm{lr} = 10^{-2}$, the curve stays stable at first and then decreases sharply until convergence, reaching a steady state faster than the curve for $\mathrm{lr} = 10^{-1}$. From Fig. 7(a), it is evident that $\mathrm{lr} = 10^{-3}$ and $\mathrm{lr} = 10^{-2}$ demonstrate comparable performance on node classification in terms of all four metrics. However, since the model with $\mathrm{lr} = 10^{-3}$ does not converge, as analyzed previously, its performance on unseen data or in different scenarios cannot be guaranteed. Therefore, we choose $\mathrm{lr} = 10^{-2}$ for rapid convergence.



(a) Classification performance with different lr.



(b) Classification performance with different k.

Figure 7: Influence of hyperparameters to classification performance.

When learning the structure of a mobile network, the nearest k data fields of one data field are considered for topology learning. The value of k influences the number of relationships that are ultimately learned. If k is too small, the model may learn too few relationships and overlook important associations, such as indirect relations in the mobile network. Conversely, if k is too large, it may acquire an excessive number of unnecessary relationships, leading to increased computational and memory demands. We set k from 1 to 10 and conducted tests to obtain multiple groups of node classification performance. The curves of accuracy, precision, recall, and F1-score with different k values are shown in Fig. 7(b). The results indicate that when k is 2, the performance of node classification is optimal. This suggests that with $k = 2$, the adjacency matrix learned is the most reasonable in terms of node classification accuracy.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 62225107, the Fundamental Research Funds for the Central Universities under Grant Nos. 2242022k60002 and 2242023R40005, and the Jiangsu Provincial Scientific Research Center of Applied Mathematics under Grant No. BK20233002.

References

Chen, Y.; Wu, L.; and Zaki, M. J. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 19314–19326. Vancouver, BC, Canada: Curran Associates Inc.

Chi, H. R.; Wu, C. K.; Huang, N.-F.; Tsang, K.-F.; and Radwan, A. 2023. A Survey of Network Automation for Industrial Internet-of-Things Toward Industry 5.0. *IEEE Transactions on Industrial Informatics*, 19(2): 2065–2077.

Fatemi, B.; El Asri, L.; and Kazemi, S. M. 2021. SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 22667–22681. Online: Curran Associates, Inc.

Franceschi, L.; Niepert, M.; Pontil, M.; and He, X. 2019. Learning Discrete Structures for Graph Neural Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, 1972–1982. Long Beach, California, USA: PMLR.

Huang, C.; Yang, Z.; Wen, J.; Xu, Y.; Jiang, Q.; Yang, J.; and Wang, Y. 2022. Self-Supervision-Augmented Deep Autoencoder for Unsupervised Visual Anomaly Detection. *IEEE Transactions on Cybernetics*, 52(12): 13834 – 13847.

Huang, Y.; You, X.; Zhan, H.; He, S.; Fu, N.; and Xu, W. 2024. Learning Wireless Data Knowledge Graph for Green Intelligent Communications: Methodology and Experiments. *IEEE Transactions on Mobile Computing*, in press, (DOI: 10.1109/TMC.2024.3408142).

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph Structure Learning for Robust Graph Neural

Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 66–74. Virtual Event, CA, USA: Association for Computing Machinery.

Liu, Y.; Zheng, Y.; Zhang, D.; Chen, H.; Peng, H.; and Pan, S. 2022. Towards Unsupervised Deep Graph Structure Learning. In *Proceedings of the ACM Web Conference 2022*, 1392–1403. Virtual Event, Lyon, France: Association for Computing Machinery.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and 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 (WSDM 2020)*, 519–527. Houston, TX, USA: Association for Computing Machinery.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Gallagher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI Magazine*, 29(3): 93–106.

van den Oord, A.; Li, Y.; and Vinyals, O. 2019. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748.

Wang, R.; Mou, S.; Wang, X.; Xiao, W.; Ju, Q.; Shi, C.; and Xie, X. 2021. Graph Structure Estimation Neural Networks. In *Proceedings of the Web Conference 2021*, 342–353. Ljubljana, Slovenia: Association for Computing Machinery.

Wang, X.; Lu, Y.; Shi, C.; Wang, R.; Cui, P.; and Mou, S. 2022. Dynamic Heterogeneous Information Network Embedding With Meta-Path Based Proximity. *IEEE Transactions on Knowledge and Data Engineering*, 34(3): 1117–1132.

Yang, L.; Xiao, Z.; Jiang, W.; Wei, Y.; Hu, Y.; and Wang, H. 2020. Dynamic heterogeneous graph embedding using hierarchical attentions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 425 – 432. Lisbon, Portugal: Springer, Cham.

You, X.; Huang, Y.; et al. 2023. Toward 6G TKμ Extreme Connectivity: Architecture, Key Technologies and Experiments. *IEEE Wireless Communications*, 30(3): 86–95.

Yu, D.; Zhang, R.; Jiang, Z.; Wu, Y.; and Yang, Y. 2021. Graph-Revised Convolutional Network. In Hutter, F.; Kersting, K.; Lijffijt, J.; and Valera, I., eds., *Machine Learning and Knowledge Discovery in Databases*, 378–393. Cham: Springer International Publishing.

Zhu, B.; Jiao, J.; and Tse, D. 2020. Deconstructing Generative Adversarial Networks. *IEEE Transactions on Information Theory*, 66(11): 7155–7179.