Modeling and Simulation 建模与仿真, 2024, 13(3), 2611-2622
Published Online May 2024 in Hans. https://www.hanspub.org/journal/mos
https://doi.org/10.12677/mos.2024.133238

Hans汉斯

基于双层时序的动态社交网络链路预测

刘　航

上海理工大学光电信息与计算机工程学院，上海

收稿日期：2024年4月20日；录用日期：2024年5月20日；发布日期：2024年5月27日

摘　要

现实世界中存在大量网络图信息，其中社交网络实体间交互的动态性和复杂性使得动态图链路预测成为了一项更具有挑战性的任务。传统基于图神经网络的动态链路方法由于过平滑性往往只关注图中局部特征，难以获取图中实体的全面性信息，并且发现在社交网络中的连通接近性对未来链路预测是有利的。为了解决以上挑战，本文设计了双层时序模型Bi-GTGNN。首先提取每个快照的子图，并将每个快照的子图集抽象为时序序列，然后设计全局时序图神经网络提取图的全局信息并生成快照表示。其次，将每个时间戳的快照表示输入到LSTM中进一步提取时序信息，并设计了新颖的损失函数训练具有连通接近性的图嵌入。最后将具有时序信息的图嵌入用于链路预测。在五个数据集上进行了大量实验，结果表示Bi-GTGNN性能优于其它先进的baseline模型。

关键词

图神经网络，动态图链路预测，双层时序，社交网络

Dynamic Social Network Link Prediction Based on Bi-Layer Temporal Modeling

Hang Liu

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai

Received: Apr. 20th, 2024; accepted: May 20th, 2024; published: May 27th, 2024

Abstract

In the real world, there is a plethora of network graph information, where the dynamism and complexity of interactions among entities in social networks have made dynamic graph link prediction a more challenging task. Traditional methods based on graph neural networks for dynamic links often focus only on local features of the graph due to over-smoothing, making it difficult to

文章引用: 刘航. 基于双层时序的动态社交网络链路预测[J]. 建模与仿真, 2024, 13(3): 2611-2622.
DOI: 10.12677/mos.2024.133238

acquire comprehensive information about entities in the graph. Additionally, we observe that the connectivity proximity in social networks is advantageous for future link prediction. To address these challenges, we propose a bi-layer temporal model. Firstly, we extract subgraphs for each snapshot and abstract the subgraph set of each snapshot into a temporal sequence. Then, we design a global temporal graph neural network to extract the global information of the graph and generate snapshot representations. Secondly, we input the snapshot representations of each timestamp into an LSTM to further extract temporal information and design a novel loss function to train graph embeddings with connectivity proximity. Finally, the graph embeddings with temporal information are utilized for link prediction. We conduct extensive experiments on five social network datasets, and the results demonstrate that our model outperforms other state-of-the-art baseline models.

Keywords

Graph Neural Network, Dynamic Graph Link Prediction, Bi-Layer Temporal, Social Network

1. 引言

现实世界的网络图数据通常会随着时间的推移而演变，社交网络平台每个月都有成千上万用户之间的链接演变，因此这些数据在社交网络[1]中得到了很好的表达。动态链路预测旨在预测节点之间未来的关系，对于理解动态图[2]的演变至关重要。现有基于图嵌入的链路预测方法可以分为传统图嵌入[3]和图神经网络图嵌入[4]。

传统的图嵌入方法包括矩阵分解和随机游走，在基于矩阵分解的技术中，节点嵌入是通过分解图表示矩阵[5]和图邻接矩阵[6]来学习的。但是，矩阵分解方法消耗大量内存空间，并且由于存储大型矩阵而需要高计算需求。因此通常通过随机游走来获取节点的邻域连通性信息。流行的基于随机行走的方法包括 deepWalk [3]和 node2vec [7]。随机行走方法侧重于保留节点接近信息，但是牺牲了网络结构信息[8]，并且高度节点可能比低度节点的样本更多[9]，因此难以广泛适用于各种网络。上述方法主要应用于静态图嵌入，目前大量研究将图嵌入技术应用于动态网络链路预测。例如，GraphLP [10]利用深度学习模型的特征学习能力，自动从图中提取结构模式，以改进链路预测。为了捕捉时间的演化信息，DynamicTriad [11]采用三元闭合的概念作为指导原则来捕捉各种快照的演化模式。DynGEM [12]从初始化开始逐步更新节点嵌入。但是难以捕获长期动态，从而限制了它们的准确性。

基于图神经网络的图嵌入方法主要通过使用图神经网络提取图中节点信息来进行链路预测。此方法最初应用于静态图，自 GCN [13]以来，许多图神经网络被提出。例如，GAT [4]、GraphSAGE [14]、FastGCN [15]、图同构网络(GIN) [16]和位置感知图神经网络(P-GNNs) [17]等。还有一些变体用于解决时空问题，通过将 GCN 变体和循环架构进行结合。例如，GCRN (Graph Convolutional recurrent Network) [18]结合了图卷积和循环神经网络的模型，在图卷积层之后引入循环神经网络层，使得模型能够在图的时间序列上进行信息传递和更新，从而捕捉快照的演化过程。AddGraph [19]结合了注意力机制和时间感知的图卷积网络，利用注意力机制来对节点的邻居进行加权，然后通过时间感知的 GCN 来聚合邻居节点的信息。但是对于复杂的图结构和时间序列数据，可能无法充分利用到演化过程中的细节信息。EvolveGCN [20]通

过引入图演化机制来建模时间演化过程，图演化机制用于动态地更新图结构和节点特征。从而能够自适应地学习图的演化规律并充分利用演化过程中的丰富信息。

上述的图神经网络由于自身架构限制，只能通过聚合低阶邻居节点信息，导致无法获取图的全局结构特征。图 1 是数据集 Enron 的部分图，如图 1 左侧红色框节点为例，当 GCN [13]行处理时，第一层 GCN 会聚合绿色填充部分的一阶邻居，此时绿色的节点也会聚合橙色节点信息。第二层 GCN 聚合时，红框节点可以通过绿色节点获取到橙色节点的信息。也就是说，红色节点想要获取二阶邻域信息必须要通过一阶邻域节点。理想情况下，当 GCN 层数无限大时，可以获取到全图信息。但是由于过平滑特性，一般情况下，当 GCN 层数达到 3 以上时，性能会极大降低。因此，传统图神经网络是无法捕获全图信息的。此外，如图 1 右侧表示时间 t 与时间 t+1 的快照变化，其中红色实线表示新增的链接，绿色虚线表示消失的链接。可以发现对于邻近的节点，在未来进行交互的可能性较大。这说明关联接近性可以帮助揭示未来社交网络中的链接变化。

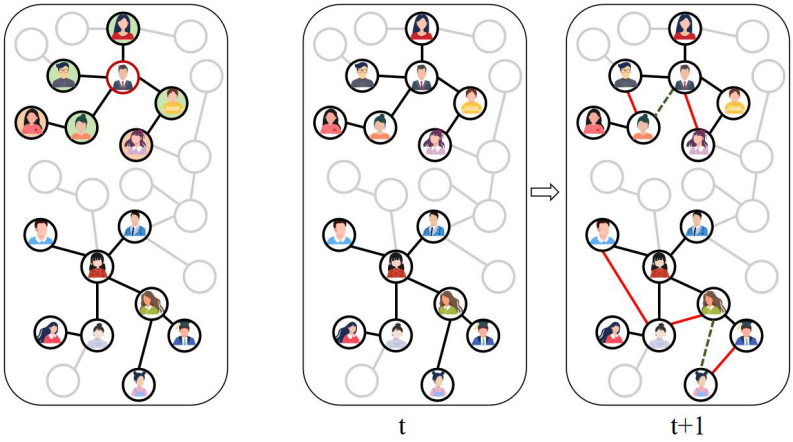

Figure 1. Problem statement
图 1. 问题阐述

为了解决以上挑战，本文提出了双层时序模型(Bi-GTGNN)，双层时序分为子图层时序以及快照层时序。子图层时序是全局时序图神经网络(Global Temporal Graph Neural Network, GTGNN)。与传统图神经网络架构不同，它不需要与低阶邻居信息聚合更新，而是对每个快照提取抽象时序子图，通过全局时序聚合获取单个快照的图嵌入表示。进一步将所有快照的图嵌入输入快照层时序捕获快照时序信息。此外，本文设计了损失函数将关联接近的用户生成相似的嵌入，以保持关联接近性。最终的嵌入用于链路预测。本文的贡献总结如下：

1) 提出了双层时序架构，子图层捕获单个快照的抽象时序子图集的时序信息，快照层捕获快照间的时序信息。

2) 传统的图神经网络只能聚合低阶邻居，难以捕获全局结构信息。本文提出的 GTGNN 模型每层都可以聚合全局信息，从而获取更丰富的隐藏特征。

3) 设计了新颖的损失函数将关联接近的用户生成相似的嵌入，从而提高预测精度。

4) 在五个真实动态社交网络数据集上进行大量实验，证明了模型的有效性。

2. 模型设计

本章介绍了提出的双层时序模型，双层时序分为子图层时序以及快照层时序。子图层时序是全局时

序图神经网络，与传统图神经网络架构不同，它不需要与低阶邻居信息聚合更新。首先提取每个时间节点快照的子图，并将这些子图抽象为一个时序演变子图集合，然后使用一个时序网络捕获抽象时序信息，这些抽象时序信息进行聚合并更新图嵌入。因此这也是一种聚合与更新的过程。快照层时序将每个时间节点快照嵌入作为 LSTM 网络的输入，获取快照时序信息。最终的嵌入用于下游链路预测任务，模型框架如图 2 所示。

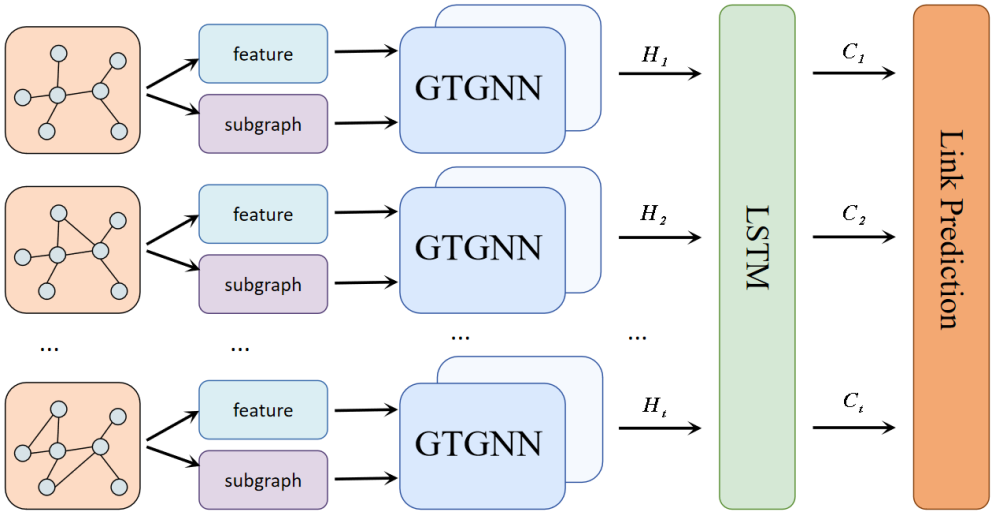

Figure 2. Bi-layer temporal framework structure
图 2. 双层时序框架结构

2.1. 定义

定义一：定义动态图为 $G(t)=(V,E)$，其中 $V=\{v_1,v_2,\cdots,v_n\}$ 代表 n 个节点集合，$E=\{E_1,E_2,\cdots,E_t\}$ 代表在 t 个时间的所有边集合。动态图可以通过一系列快照 $(G_1,G_2,\cdots,G_t)$ 来表示，G_t 代表图在时间点 t 的快照状态。每个时间点的快照邻接矩阵表示为 A_t，如果节点 v_i 和 v_j 之间无连接，则邻接矩阵 $A_{i,j}=0$，否则为 $A_{i,j}>0$。图嵌入方法学习映射 $F=\{f_1,f_2,\cdots,f_t\}$，其中 $f^t\in F$ 将图 G_t 中的每个节点 v_n 映射到向量空间 R^d，d 表示节点表示维度，节点嵌入可以表示为 X。映射后的节点表示可以保留节点的拓扑信息以及动态图的时序信息。

定义二：将每一个快照生成子图集，子图集可以表示为 $\tilde{G}=\{\tilde{G}^1,\tilde{G}^2,\cdots,\tilde{G}^n\}$。$\tilde{G}^n$ 表示当前子图每个节点度大于等于 n，因此 $\tilde{G}^1$ 也表示原图。子图的生成过程可以表示为，给定图 $G=(V,E)$，通过递归移除度小于 n 的节点直到所有节点的度都至少为 n。

2.2. 子图层时序

2.2.1. 子图提取

子图层时序需要先提取快照子图，本文中的子图提取可视为一种图划分方法，类似的方法已在 ClusterGCN [21]中使用。ClusterGCN 通过图聚类算法将图划分为多个子图，以限制 GCN 层中的邻域扩展，从而解决 GCN 中的过度平滑问题。本文目标是增强图神经网络，以保留局部连接近似，并在动态图中捕获全局结构信息。如图 3 所示，这是一个简单的子图提取示例，可以发现，$\tilde{G}^3$、$\tilde{G}^2$ 和 $\tilde{G}^1$ 中节点数量相同，但是边不同。并且由于是通过对原图的边进行移除，它们之间存在着一种抽象时序关系。如：$\tilde{G}^3\to\tilde{G}^2\to\tilde{G}^1$。

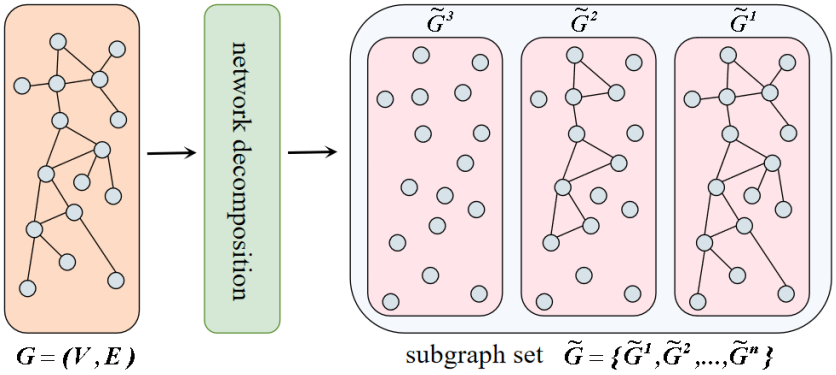

Figure 3. Example of subgraph extraction process
图 3. 子图提取过程示例

2.2.2. 全局时序图神经网络

图卷积网络(GCNs)在各种任务中取得了较好的表现。形式上，一个单层的普通 GCN 被定义为：

$$H^{(l+1)}=\sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{1}$$

其中 X 是输入特征矩阵，$\tilde{A}=A+I$ 表示具有自环的邻接矩阵，I 是单位矩阵，$\tilde{D}$ 是对角矩阵，其元素 $\tilde{D}_{ij}=\sum_j\tilde{A}_{ij}$，$W$ 是可学习的权重矩阵，$\sigma(\cdot)$ 是非线性激活函数，H 表示学习到的输出嵌入。

显然，图卷积层类似于全连接层，除了扩散矩阵 $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ 外，它将节点的邻居特征传播到该节点。本质上，图卷积是一种特殊形式的拉普拉斯平滑，它计算一个节点的新特征作为自身及其邻居的加权平均。单层 GCN 可以保留一阶接近性，而多层 GCN 在实践中经常使用，因为它可以捕获高阶接近性信息。然而，由于图神经网络架构的过平滑性，当图的卷积层增加时，GCN 会出现过平滑和过拟合现象。这些问题反映了 GCN 无法探索全局图结构。因此，在实践中，一个 GCN 通常最多包含 3 个图卷积层。

受此启发，本文将图神经网络分为特征变换和特征聚合，可以如下表达：

$$Z=f(X) \tag{2}$$

$$H=h\left(g(A),Z\right) \tag{3}$$

其中，X 为输入特征矩阵，f 为将 X 映射到潜空间的特征变换函数，A 为邻接矩阵，g 为定义传播规则的 A 的函数，h 为根据传播规则 g 对每个节点的特征进行聚合的特征聚合函数，h 是融合了特征变换和图结构信息的嵌入矩阵。

许多以前的工作都是不同的 f 和 g 组合，但它们都选择 H 的输出作为实际的最终嵌入。本文将 f 和 H 的输出都用作节点表示，这将在后面讨论。通过将传统的图卷积运算推广到特征变换和特征聚合，为设计不同类型的 GCNs 提供了更大的灵活性。接下来介绍 GTGNN，结构如图 4 所示。

前面介绍的抽象时序子图集可以表示为时间序列 $\tilde{G}^n\to\tilde{G}^{n-1}\to\cdots\to\tilde{G}^2\to\tilde{G}^1$。本文使用的是每个子图的邻接矩阵作为输入，然后与初始节点嵌入进行特征变换，如公式(4)

$$X^n=\sigma\left(\tilde{A}^nX\right) \tag{4}$$

其中，$\tilde{A}^n$ 是 $\tilde{G}^n$ 子图的自环邻接矩阵，也就是在邻接矩阵上增加了一个单位矩阵。由于将子图集抽象为时间序列，因此特征聚合将在提取了时间信息后进行。由于时间序列 $\tilde{G}^n\to\tilde{G}^{n-1}\to\cdots\to\tilde{G}^2\to\tilde{G}^1$ 的表示是从 $n\to1$，为了后续表达清晰，本文在后续将其表示为 $\tilde{G}^1\to\tilde{G}^2\to\cdots\to\tilde{G}^{n-1}\to\tilde{G}^n$，例如 $\tilde{G}^1$ 的真实含

义是 $\tilde{G}^n$，就是节点度大于等于 n 的图。$\tilde{G}^n$ 的真实含义是 $\tilde{G}^1$，就是节点度大于等于 1 的图，等同于原图。在图 4 中，也是进行了相关调整。

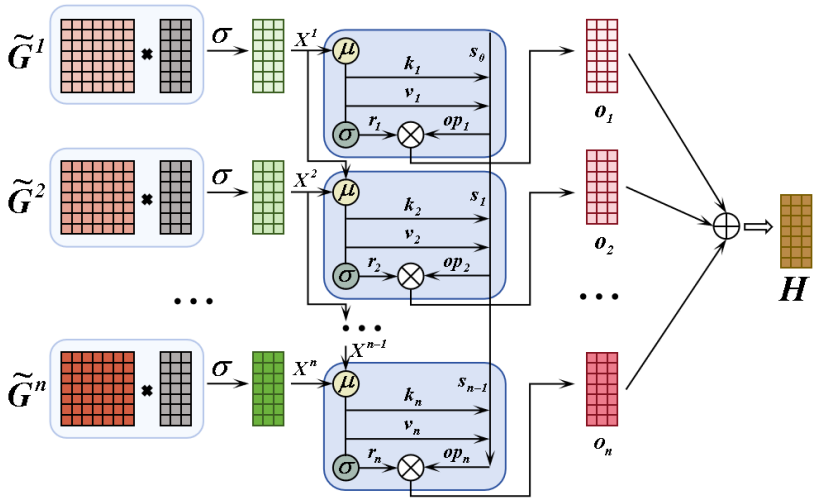

Figure 4. GTGNN structure
图 4. GTGNN 结构

现有的 RNN 架构是比较常见的时序方法，但是这些方法难以捕获长序列信息并且受限于对前一个时间序列数据的依赖，因此时间复杂度较大并且性能不佳。本文使用了时间混合方法调整了 RNN 架构，因为 RNN 依赖于前一个时间片的数据信息和隐层表示。将前一个时间片 $t-1$ 的数据信息与当前时间片 t 的数据信息进行混合操作，并将隐层嵌入使用线性表示，则可以直接获取长序列信息并减少时间复杂度。

时序混合如公式(5) (6) (7)所示，μ_r，μ_k，μ_v 用于 X^n 和 X^{n-1} 的线性组合中，以实现简单的时间混合，这种混合在当前和前一时间步的输入之间进行插值。前一步和当前步的组合通过块内的投影矩阵进行线性投影。

$$r_n = W_r \cdot \left(\mu_r \odot X^n + \left(1-\mu_r\right) \odot X^{n-1}\right) \tag{5}$$

$$k_n = W_k \cdot \left(\mu_k \odot X^n + \left(1-\mu_k\right) \odot X^{n-1}\right) \tag{6}$$

$$v_n = W_v \cdot \left(\mu_v \odot X^n + \left(1-\mu_v\right) \odot X^{n-1}\right) \tag{7}$$

进一步设计一个类似于注意力机制的操作器，它具有线性时间和空间复杂度。也可以认为是对隐层嵌入的操作器。公式如下：

$$op_n = diag\left(u\right) \cdot k_n^T \cdot v_n + \sum_{i=1}^{n-1} diag\left(w\right)^{n-1-i} \cdot k_i^T \cdot v_i \tag{8}$$

其中 $\sum_{i=1}^{n-1} diag\left(w\right)^{n-1-i} \cdot k_i^T \cdot v_i$ 可以认为是上一层的隐层嵌入，w 和 u 是两个可训练参数。参数 u 是一个奖励参数，首次遇到一个 token，特别是当前 token 的奖励。这有助于模型更加关注当前 token。另一个重要的参数是 w，它是每个时间片隐藏表示的时间衰减向量。为了确保对前面每个时序通道的隐藏表示进行惩罚，需要将 w 表示在(0, 1)范围，因此对 w 的处理如公式(9)。

$$w = \exp\left(-\exp\left(w\right)\right) \tag{9}$$

由于可以直接获取 $k_n^T \cdot v_n$，因此每个时序通道可以直接进行计算。每个时序通道的输出表示如公式(10)

$$o_n = \left(SiLU\left(r_n\right) \odot op_n\right) W_{op} \tag{10}$$

上述过程可以以 RNN 循环形式进行表达，如公式(11) (12)：

$$op_n = s_{t-1} + diag\left(u\right) \cdot k_n^T \cdot v_n \tag{11}$$

$$s_t = diag\left(w\right) \cdot s_{t-1} + k_n^T \cdot v_n \tag{12}$$

其中s_t为当前时间步隐层状态，s_{t-1}为上一时间步隐层状态，s_0为零矩阵。最终获得了具有时序信息的输出$O = \left\{o_1, o_2, \cdots, o_n\right\}$。

完成对子图集中每个子图的嵌入工作后，接下来需要将每个子图的嵌入表示进行聚合，如公式(13)，由此完成一次完整的特征变换与聚合。对于下一层的输入，则由上一层的输出 H 与子图集的自环邻接矩阵进行特征变换并重复上述操作。

$$H = \sum_{l=1}^{n} o_l \tag{13}$$

2.3. 快照层时序

将每个单一快照使用 GTGNN 进行两次处理后。会得到每个快照的节点特征矩阵集合$\left\{H_1, H_2, \cdots, H_t\right\}$。然后将这 t 个节点特征矩阵作为序列输入到 LSTM 模型中。就可以获取到具有快照层时态信息的最新的t个节点特征矩阵。具体公式如下：

$$C_t = \mathrm{LSTM}\left(\mathrm{GTGNN}^t\left(A_t, X\right), C_{t-1}\right) \tag{14}$$

其中C_0为零矩阵，A_t为G_t的邻接矩阵，X为 G 的节点特征矩阵。每个 GTGNN 对 LSTM 单元产生静态节点嵌入，LSTM 单元决定下一个时间戳层保留多少信息。节点特征矩阵$C_1, C_2, \cdots, C_t$是 t 个快照的最终嵌入。

2.4. 损失函数

在引言部分已经阐述了连通接近性对于链路预测的影响。为了在动态图中保持连接的接近性，本文遵循前面的 GCNs 来利用 h 函数的输出作为最终的嵌入。模型训练采用无监督方式，损失函数如下：

$$L_c = \sum_{t=1}^{T} \sum_{u \in V} L_u^t \tag{15}$$

$$L_u^t = \sum_{v \in N_w^t(u)} -\log\left\{1 - \sigma\left[D_{(h_v, h_u)}\right]\right\} - Q \cdot \sum_{v' \in P_n^t(u)} -\log\left\{\sigma\left[D_{(h_{v'}, h_u)}\right]\right\} \tag{16}$$

$$D_{(h_v, h_u)} = \sqrt{\sum_{i=1}^{d} \left(h_{vi} - h_{ui}\right)^2} \tag{17}$$

其中$D_{(h_v, h_u)}$表示欧式距离，d 是嵌入维度，$N_w^t\left(u\right)$是与 u 在定长随机游动上同时出现的节点集合，P_n^t 表示负样本集合。Q 是用于平衡正、负样本的超参数。通过这个无监督的损失函数，在固定长度的随机游动中发生的节点被鼓励有相似的表示。因此，学习后的节点表示可以在动态图中保持局部连接的邻近性。

3. 实验结果与分析

3.1. 数据集与评估指标

实验中使用了五个社交网络数据集。包括加州大学欧文分校的一个学生信息网络 UCI；来自

Autonomous systems 的一个通信网络 AS；来自 Facebook 公司的 Facebook 数据集；一个来自安然公司的电子邮件联系网络 Enron 和一个来自堆栈交换网站 Math overflow 的用户交互网络 Math。在实践中按月分割数据集。数据集信息如下表 1。

Table 1. Dataset information
表 1. 数据集信息

Datasets	Nodes	Edges	D_{max}	Snapshots
UCI	1899	59835	198	7
AS	6828	1,947,704	1458	100
Math	24,740	323,357	231	77
Facebook	60,730	607,487	203	27
Enron	87,036	530,284	1150	38

评估指标使用的是曲线下面积(Area Under The Curve, AUC)，ROC 曲线(Receiver Operating Characteristic curve)下方的面积就是 AUC。因为最终的链路预测任务是二分类问题，而 AUC 的优点在于它对类别不平衡和阈值选择不敏感，因此被广泛应用于评估各种分类任务的性能。

3.2. 对比模型和实施细节

对比 baseline 分为静态图方法和动态图方法，静态图方法选择 GCN [13]、GAT [4]和 GIN [16]，动态图方法选择 DynGEM [12]、Dyngraph2vec [22]、GCRN [18]和 EvolveGCN [20]。其中 Dyngraph2vec 有三种变体，分别是 dynAE、dynRNN 和 dynAERNN。

对于基于静态 GCN 的方法，在 GCN、GAT、GIN、GCRN 和 EvolveGCN 中使用了 2 层。对于 DynGEM，将 α 设为 10^{-5}，β 设为 10，γ_1 设为 10^{-4}，γ_2 设为 10^{-4}。对于 dynAE、dynRNN 和 dynAERNN，设 β 为 5，向后看为 3，γ_1 为10^{-6}，γ_2 为10^{-6}。对于本文所提出的方法 Bi-GTGNN，为了公平对比，也保持两层 GTGNN。优化器使用 Adam [23]，学习率设置为 0.001。为了方便比较，所有方法的嵌入维度都设置为 128。

给定一系列动态图，链接预测根据当前时间步长 t 的嵌入信息，预测下一个时间步长 $t+1$ 中存在一条边。为了构建带标记的边集，本文从图 G_t 中随机采样边作为正样本，从未连接的节点对中生成负样本。然后采用[7]的方法，使用标记的边集中节点对的嵌入向量执行 Hadamard 运算以生成边特征向量。训练一个逻辑回归(Logistic Regression, LR)分类器来区分正和负边样本。所有静态图嵌入方法都在每个图快照上重新训练以生成节点嵌入，而所有动态图嵌入方法都结合了历史图信息以生成当前时间戳的节点嵌入。

3.3. 对比实验结果分析

如表 2 所示，实验结果是所有时间戳的平均 AUC 分数，“-”表示 GPU 内存不足。实验结果表示静态图方法整体性能低于动态图方法，因为常规的图嵌入方式提取的动态图信息存在缺陷。说明时序信息在动态图链路预测中是不可或缺的。动态嵌入方法中，Bi-GTGNN 性能优于其它方法，这表明 Bi-GTGNN 可以捕获全局图结构信息，从而生成的节点表示有助于预测链路的形成和消失。其它动态图嵌入方法表现最优的是 EvolveGCN，这是因为它引入了图演化机制，能够根据当前时刻的图数据动态地更新节点表示。这使得模型能够更好地捕捉动态图的时间演化规律，从而提高了嵌入质量。

Table 2. Comparative experimental results
表 2. 对比实验结果

Method	UCI	AS	Math	Facebook	Enron
GCN	0.7792	0.7835	0.7986	0.6942	0.7879
GAT	0.7668	0.7906	0.8351	0.6991	0.8344
GIN	0.8366	0.8571	0.8681	0.7415	0.8453
DynGEM	0.9032	0.9372	0.9025	0.8019	0.8926
dynAE	0.9189	0.9404	0.9178	-	-
dynRNN	0.8927	0.9107	-	-	-
dynAERNN	0.9008	0.9139	-	-	-
GCRN	0.8579	0.8648	0.8217	0.7262	0.8807
EvolveGCN	0.9102	0.9227	0.9034	0.8056	0.9025
Bi-GTGNN	0.9214	0.9436	0.9279	0.8591	0.9307

3.4. 消融实验

消融实验继续在 5 个社交网络数据集上进行，Baseline 模型使用 GCN、GAT、GIN。在之前的对比实验中，对于这三个 baseline 的操作方式是分别对每个快照生成节点表示，然后进行链路预测。在消融实验中，在分别生成节点表示后，再输入到快照层时序进一步生成新的表示，损失函数如公式(15)，最后进行链路预测。调整后的模型表示为 GCN++、GAT++和 GIN++。此外，本文增加了 GTGNN 的实验，并且训练方式与未融合快照层的 GCN 相同。

如表 3 所示，GCN、GAT 和 GIN 融合快照层时序后，在 5 个数据集上都有了一定提升，这说明快照层时序是有效的。其中，在 AS 数据集上各个模型提升较大，这是因为 AS 数据集中快照数量最多，因此快照间的时序信息是非常丰富的，而 GCN、GAT 和 GIN 都无法捕获快照间时序。加入快照层后，它们都可以生成具有时序信息的节点表示。此外，使用公式(15)的损失函数后，它们生成的嵌入具有连通接近性，从而进一步提高预测性能。

Table 3. Ablation experiment results
表 3. 消融实验结果

Method	UCI	AS	Math	Facebook	Enron
GCN	0.7792	0.7835	0.7986	0.6942	0.7879
GCN++	0.8204	0.8357	0.8341	0.7312	0.8134
GAT	0.7668	0.7906	0.8351	0.6991	0.8344
GAT++	0.8079	0.8311	0.8653	0.7251	0.8570
GIN	0.8366	0.8571	0.8681	0.7415	0.8453
GIN++	0.8615	0.8762	0.8807	0.7849	0.8701
GTGNN	0.8535	0.8763	0.8719	0.7963	0.8832
Bi-GTGNN	0.9214	0.9436	0.9279	0.8591	0.9307

GCN、GAT 和 GIN 的性能弱于 GTGNN，这说明 GTGNN 解除了只能聚合低阶邻居的限制并充分考虑全局信息是有效的。此外，GCN++、GAT++和 GIN++性能也弱于 Bi-GTGNN，因为即使 GCN、GAT 和 GIN 融合了快照层时序信息，但由于过平滑性限制了对远距离节点探测，所以生成的节点表示信息不足，因此预测性能难以提升。

3.5. 并行性分析

在 GTGNN 中有一个时序模块，本文在 3.2.2 节中阐述了每个时间戳的计算并行性，本节将分析并行原理以及实验结果。

首先常规的 RNN 模型每个时间通道需要使用上一个时间通道的隐层输出作为当前层的输入，因此只能保持串行进行计算。在 GTGNN 中的时序部分，如公式(10)，每个时间通道的输出由 r_n 和 op_n 计算得出，其中 r_n 是已知的。如公式(8)，op_n 则由当前时间通道 k_n 、v_n 与之前所有时间通道的 k、v 计算得出，而所有的 k、v 是已知的。因此任意层的计算所需变量都可以直接获得，从而可以实现并行计算。

在数据集 Facebook 和 Enron 上对比了 GCN、GAT、GIN、Bi-GTGNN 和 Bi-GTGNN-R 训练每个 epoch 所花费的时间，其中 Bi-GTGNN-R 是使用 RNN 替换 GTGNN 中的时序模块，其它实验设置与对比实验相同。如图 5 所示，Bi-GTGNN-R 的计算效率将近高于 GTGNN 的一倍，说明 GTGNN 中的并行计算是有效的。GAT 的训练时间最高，远超于其它方法，训练时间最低的是 GCN。这是因为 GAT 在每次卷积融合特征的过程中引入了注意力机制。这个机制允许模型为每条边分配一个可学习的权重系数 α，使得特征融合过程更加灵活和自适应。具体来说，GAT 的每个节点都会计算与其邻居节点之间的注意力系数，这个过程涉及到额外的参数和非线性变换，如 LeakyReLU 函数。而且，GAT 通常采用多头注意力机制，即在同一层中并行计算多组注意力系数，然后将它们合并起来，以增强模型的表达能力，因此训练时间远大于 Bi-GTGNN。相比之下，GCN 的卷积过程相对简单，它使用度矩阵和邻接矩阵来计算邻居节点的加权平均，而这些权重是固定的，不涉及额外的学习过程。因此，GCN 的计算效率更高，训练时间较短。

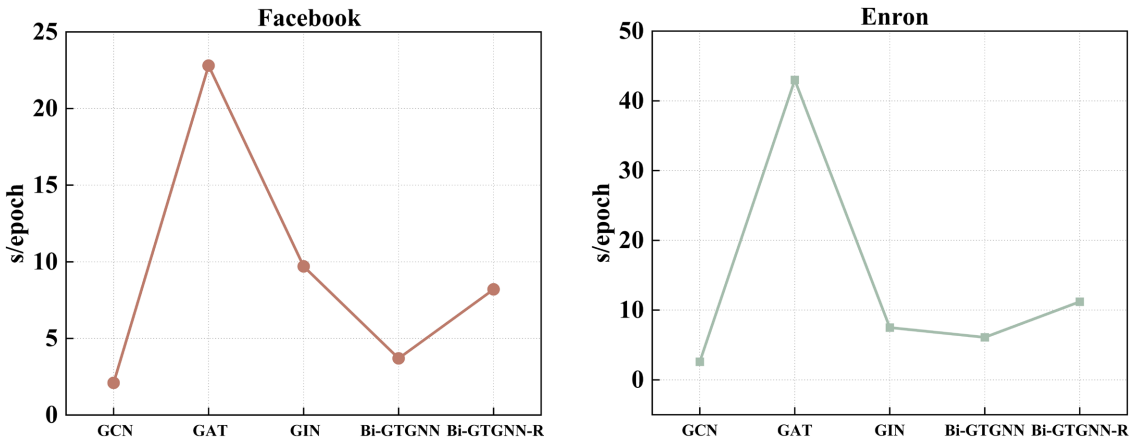

Figure 5. Training time results
图 5. 训练时间结果

4. 总结

本文提出了 Bi-GTGNN 模型。Bi-GTGNN 模型是一个双层时序模型，子图层时序通过提取抽象时序子图集并聚合信息，从而可以充分获取图的全局信息。快照层时序使用 LSTM 捕获每个时间戳的快照间时序，并设计了损失函数训练具有连通接近性的节点嵌入。最终的嵌入用于链路预测任务。在五个不同

的社交网络数据集上实验比较了 Bi-GTGNN 与其它的基线方法，实验结果证明 Bi-GTGNN 具有先进的性能。其次将双层时序对基线方法进行了扩展以证明每个模块的有效性。最后分析了 GTGNN 中时序模块的并行性。在未来，可以探索使用更高效的算法来优化快照层时序，例如通过简化神经网络结构或采用轻量级的变体。同时，对于模型的训练过程，可以采用分布式计算资源，以进一步缩短训练时间并提升模型的响应速度。在保证模型准确性的前提下，这些改进将有助于模型在实际应用中的部署和运行。

参考文献

[1] Yang, X., Yang, Y., Su, J., *et al*. (2022) Who's Next: Rising Star Prediction via Diffusion of User Interest in Social Networks. *IEEE Transactions on Knowledge and Data Engineering*, **35**, 5413-5425. https://doi.org/10.1109/TKDE.2022.3151835

[2] Yang, C., Wang, C., Lu, Y., *et al*. (2022) Few-Shot Link Prediction in Dynamic Networks. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, New York, 21-25 February 2022, 1245-1255. https://doi.org/10.1145/3488560.3498417

[3] Perozzi, B., Al-Rfou, R. and Skiena, S. (2014) DeepWalk: Online Learning of Social Representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, 24-27 August 2014, 701-710. https://doi.org/10.1145/2623330.2623732

[4] Veličković, P., Cucurull, G., Casanova, A., *et al*. (2017) Graph Attention Networks. arXiv:1710.10903. https://doi.org/10.48550/arXiv.1710.10903

[5] Yang, C., Liu, Z., Zhao, D., *et al*. (2015) Network Representation Learning with Rich Text Information. *Proceedings of the 24th International Conference on Artificial Intelligence*, Buenos Aires, 25-31 July 2015, 2111-2117.

[6] Gligorijević, V., Panagakis, Y. and Zafeiriou, S. (2018) Non-Negative Matrix Factorizations for Multiplex Network Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **41**, 928-940. https://doi.org/10.1109/TPAMI.2018.2821146

[7] Grover, A. and Leskovec, J. (2016) node2vec: Scalable Feature Learning for Networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, 13-17 August 2016, 855-864. https://doi.org/10.1145/2939672.2939754

[8] Xia, F., Liu, J, Nie, H., *et al*. (2019) Random Walks: A Review of Algorithms and Applications. *IEEE Transactions on Emerging Topics in Computational Intelligence*, **4**, 95-107. https://doi.org/10.1109/TETCI.2019.2952908

[9] Kojaku, S., Yoon, J., Constantino, I., *et al*. (2021) Residual2Vec: Debiasing Graph Embedding with Random Graphs. arXiv:2110.07654. https://doi.org/10.48550/arXiv.2110.07654

[10] Xian, X., Wu, T., Ma, X., *et al*. (2022) Generative Graph Neural Networks for Link Prediction. arXiv:2301.00169. https://doi.org/10.48550/arXiv.2301.00169

[11] Zhou, L., Yang, Y., Ren, X., *et al*. (2018) Dynamic Network Embedding by Modeling Triadic Closure Process. *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, 2-7 February 2018. https://doi.org/10.1609/aaai.v32i1.11257

[12] Goyal, P., Kamra, N., He, X., *et al*. (2018) Dyngem: Deep Embedding Method for Dynamic Graphs. arXiv:1805.11273. https://doi.org/10.48550/arXiv.1805.11273

[13] Kipf, T.N. and Welling, M. (2016) Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907. https://doi.org/10.48550/arXiv.1609.02907

[14] Hamilton, W., Ying, Z. and Leskovec, J. (2017) Inductive Representation Learning on Large Graphs. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 1025-1035.

[15] Chen, J., Ma, T. and Xiao, C. (2018) Fastgcn: Fast Learning with Graph Convolutional Networks via Importance Sampling. arXiv:1801.10247. https://doi.org/10.48550/arXiv.1801.10247

[16] Xu, K., Hu, W., Leskovec, J., *et al*. (2018) How Powerful Are Graph Neural Networks? arXiv:1810.00826. https://doi.org/10.48550/arXiv.1810.00826

[17] You, J., Ying, R. and Leskovec, J. (2019) Position-Aware Graph Neural Networks. *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, 9-15 June 2019, 7134-7143.

[18] Seo, Y., Defferrard, M., Vandergheynst, P., *et al*. (2018) Structured Sequence Modeling with Graph Convolutional Recurrent Networks. *Neural Information Processing*: 25*th International Conference*, Siem Reap, 13-16 December 2018, 362-373. https://doi.org/10.1007/978-3-030-04167-0_33

[19] Zheng, L., Li, Z., Li, J., *et al*. (2019) AddGraph: Anomaly Detection in Dynamic Graph Using Attention-Based Temporal GCN. *Proceedings of the* 28*th International Joint Conference on Artificial Intelligence*, Macao, 10-16 August 2019, 4419-4425. https://doi.org/10.24963/ijcai.2019/614

[20] Pareja, A., Domeniconi, G., Chen, J., *et al*. (2020) EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**, 5363-5370. https://doi.org/10.1609/aaai.v34i04.5984

[21] Chiang, W.L., Liu, X., Si, S., *et al*. (2019) Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage, 4-8 August 2019, 257-266. https://doi.org/10.1145/3292500.3330925

[22] Goyal, P., Chhetri, S.R. and Canedo, A. (2020) dyngraph2vec: Capturing Network Dynamics Using Dynamic Graph Representation Learning. *Knowledge-Based Systems*, **187**, Article 104816. https://doi.org/10.1016/j.knosys.2019.06.024

[23] Kingma, D.P. and Ba, J. (2014) Adam: A Method for Stochastic Optimization. arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980