NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

Wenchao Yu, Wei Cheng, Charu C. Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang
Department of Computer Science, University of California Los Angeles
NEC Laboratories America, Inc. IBM Research AI
Department of Computer and Information Sciences, Temple University

摘要

大规模的动态网络存在于许多实际应用中，如社交媒体、安全和公共卫生等等领域。对于一个动态网络，实时检测结构异常是至关重要的，比如其“行为”偏离网络底层大多数的顶点和边。随着深度学习技术的发展，网络嵌入已经被证明是学习网络中顶点的低维表示的一个强大的工具，它可以捕获和保存网络结构。然而，大多数现有的网络嵌入方法都是为静态网络设计的，因此可能并不完全适合网络表示必须不断更新的动态环境。NetWalk 方法通过学习网络表示来进行动态网络异常检测，并且网络表示可以随着网络的发展而动态更新。首先通过团嵌入的方式将动态网络的顶点编码为向量表示，这将最小化动态网络中每个行走顶点表示的成对距离，并将深度自编码器重构误差作为全局正则化。矢量表示可以通过储层采样在恒定空间要求下计算。在学习到的低维顶点表示的基础上，采用基于聚类的技术增量动态检测网络异常。

关键词：异常检测；动态网络嵌入；深度自编码器；团嵌入

1 引言

随着互联网和社会的快速发展，网络和现实生活中的数据和规模与日俱增，数据之间的关系也因此变得愈加复杂。动态图是多张图按时间顺序排列成的图序列，是表示对象、对象之间关系和它们如何随着时间变化的一种有效方法。与图不同，动态图中图的结构或节点属性会随着时间的推移而变化，例如发生节点、边的添加和删除，或发生节点属性的变化等。由于对象以及对象之间的关系可以用动态图有效地表示，动态图技术被广泛应用各种领域，例如社交网络、计算机网络、分子生物学等。

动态变化网络中的异常检测 (又称离群点检测) 是一个长期存在的问题，在许多应用领域都有着至关重要的作用。异常指的是与标准、预期模式不同的对象。例如，邮件系统中的垃圾邮件，计算机网络中的网络攻击或其他异常流量，社交网络中的诈骗信息，这些都是生活中比较常见的异常。在很多情况下，异常会带来负面影响。例如，社交网络中，犯罪分子传播欺诈信息，从而欺骗网络中的用户并非法牟利；在计算机网络中，黑客攻击计算机主机，窃取或破坏计算机的重要资料，造成经济损失。许多应用需要异常检测，以避免异常带来的负面影响。异常检测是数据挖掘的过程，旨在识别数据集中偏离标准、预期模式的对象。近年来，异常的负面影响日益增加，因此受到越来越多研究人员的关注。在各种应用中，异常检测的需求越来越大。

传统的图异常检测方法大多是基于启发式的，缺点是需要人工设计特征，导致成本大，灵活性不高。由于异常是灵活多变的，在大规模的动态图上，需要人工设计特征的方法难以有好的性能。随着

深度学习技术的发展，基于图表示学习的网络嵌入引起了人们的极大兴趣，其可以将节点映射到一个低维向量空间，并保留图的内在属性，网络嵌入的结构保留特性使其特别适合于异常检测任务，通过检查潜在表示中顶点/边之间的相似性。例如，在多维潜在空间中，远离大多数聚类的顶点很有可能表明某些类型的异常，通过动态聚类算法可以方便地检测到这些异常。其次，现有的网络嵌入方法由于不断有新的顶点或边输入，不能动态更新表示，因此可能并不完全适合动态环境中的异常检测。在快速发展的网络中，这个问题可能更具挑战性。因此，设计一个有效的，特别是高效的嵌入算法是非常可取的，它能够在有限的内存使用下实现对异常的快速实时检测。

2 相关工作

在多维数据和结构化网络的背景下，异常检测已被广泛应用。大量的网络出现在许多应用中，如社交媒体和公共卫生，因此已经开发了许多算法来处理数据流模型中的网络。在本节中，我们简要回顾了动态网络中的异常检测算法，以及网络嵌入技术。

2.1 动态图中的异常检测

GOutlier[1]设计了一种蓄水池采样方法来维护动态图的结构信息，基于这些信息，启发式地定义异常的规则来找到异常。CM-Sketch[2]利用草图技术来提供恒定的时间和空间复杂度，并基于草图，提取全局和局部结构特征，在提取到的特征上计算异常分数。StreamSpot[3]设计了基于局部子结构的相似函数，接着使用聚类算法区分正常和异常数据。GMicro[4]通过使用基于哈希后的边从图流创建微簇，可以减少表示的大小，并保持方法的性能。SpotLight[5]通过随机抽样节点集和计算节点集和当前边集节点之间的重叠，将原有动态图映射到一个新的草图空间中，在该草图空间中，异常图远离正常图。接着利用草图空间中的数据之间的距离来检测异常。上述方法使用启发式规则来定义动态图的特征。然而，异常的模式是多变和复杂的。启发式规则在适应复杂、灵活多变的异常模式方面具有挑战性。

2.2 图嵌入技术

基于图表示学习的方法可以将节点映射到低维空间，并保留图的内在属性，灵活性更高，性能表现更好。在图表示学习技术中，DeepWalk[6]，LINE[7]，node2vec[8]，struc2vec[9]使用不同的采样算法在图上采样随机游走，并应用词向量技术学习节点嵌入。在面向动态图的图表示学习技术中，许多工作在图的随机游走方法基础上，扩展了随机游走算法到动态图中。例如，Dynnode2vec[10]在图更新时，没有对整个图进行重新采样，而是在有变化的节点上采样随机游走，用新的随机游走来训练网络。然而，这些图表示学习方法并不专门面向异常检测设计，在异常检测领域效果不佳。因此，出现了一些在异常检测领域上面向动态图的图表示学习方法。AddGraph[11]使用图卷积神经网络、注意力和 GRU 分别提取动态图的结构特征和时序特征，并引入异常得分层进行异常检测。基于图表示学习技术且面向动态图的异常边检测方法，能够更好地提取特征，适应复杂的异常模式，其异常检测的性能也优于传统基于启发式规则的方法。

3 本文方法

3.1 NetWalk 方法概述

NetWalk 会随着网络的发展，逐步学习网络表示，并实时检测网络中的异常。首先，NetWalk 使用从初始网络中提取的大量网络游走来学习潜在网络表示。该表示方法不仅通过在局部行走中保持对顶

点距离，而且通过与深层自编码器的隐层杂交得到，从而保证所得到的嵌入能忠实地重建原始网络。通过这种方法，学习到的多维欧氏空间顶点坐标既能实现局部拟合又能实现全局正则化。此外，通过利用储层采样策略，可以很容易地根据动态变化更新学到的表示形式。然后，基于学习到的顶点或边缘表示，使用动态聚类模型标记异常顶点或边缘。下图 1是动态网络异常检测的 NetWalk 工作流程：

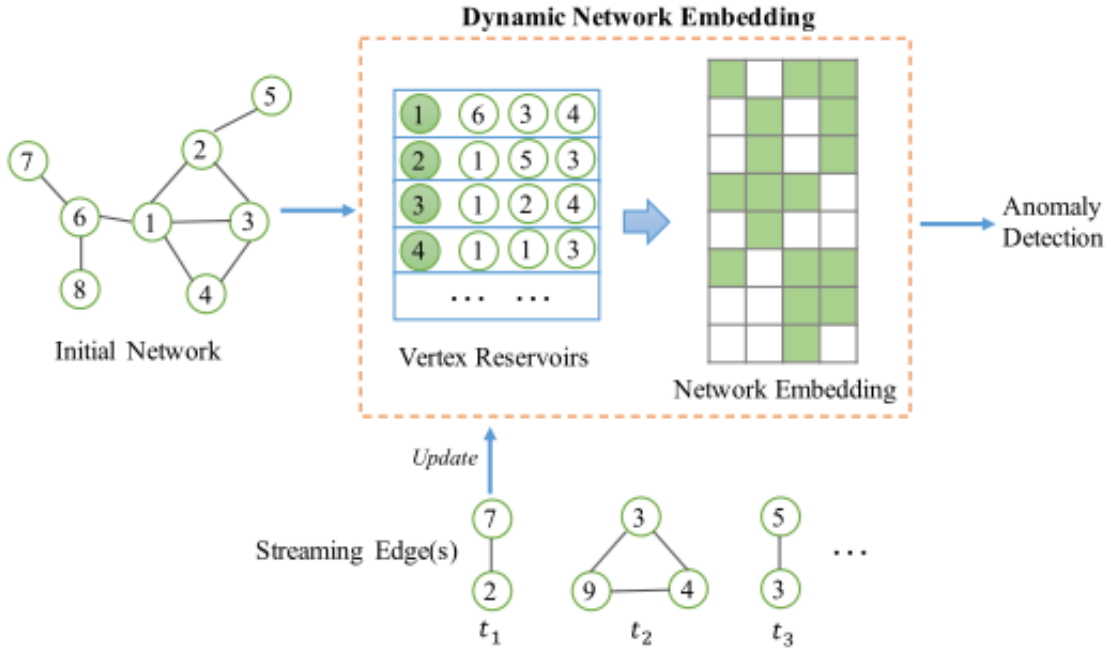

图 1: 动态网络异常检测的 NetWalk 工作流程

3.2 网络表示学习

为了实时检测动态网络中的异常，NetWalk 方法需要学习网络表示，并随着网络的发展有效地进行在线更新。本节主要介绍了 NetWalk 中的网络编码和更新阶段，它不需要存储整个网络。

3.2.1 生成网络随机游走

类似于构建向量表示时的单词嵌入技术 [25,26]，NetWalk 将网络分解为一组网络行走，每组网络行走都包含由随机行走选择的顶点列表，将使用网络行走流作为从网络中提取信息的基本工具。网络游走正式的定义如下：

定义 3.1 (网络游走)。对于网络 $\mathcal{G}(\mathcal{E},\mathcal{V})$ 中给定顶点 $v_1 \in \mathcal{V}$，其网络游走集定义为

$$\Omega_{v_1} = \{(v_1, v_2, \ldots, v_l) \mid (v_i, v_{i+1}) \in \mathcal{E} \wedge p\,(v_i, v_{i+1}) = \frac{1}{D_{v_i,v_i}}\}$$

这是从顶点 v_1 开始的 $l-hop$ 行走的集合。从 v_i 到 v_{i+1} 的转移概率 $p(v_i, v_{i+1})$ 与顶点 v_1 的度数 $D_{i,i}$ 成正比。我们称 Ω_v 为从 v 开始的网络游走集，$\Omega = \{\Omega_v\}_{v\in\mathcal{V}}$ 为所有游走集的并集。

3.2.2 节点表示学习

NetWalk 将图的表示学习问题表述为一个优化问题，其目标是学习一个映射函数 $f : \mathcal{V} \to \mathbb{R}^d$，使每个节点 $v \in \mathcal{V}$ 表为一个 d 维向量，其中 d 是空间维数。映射函数 f 适用于任何无向、有向、不带权、带权网络。NetWalk 提出了一种新的嵌入算法——团嵌入，图 2 以长度为 3 的网络行走为例，描述了整个团嵌入具体过程。

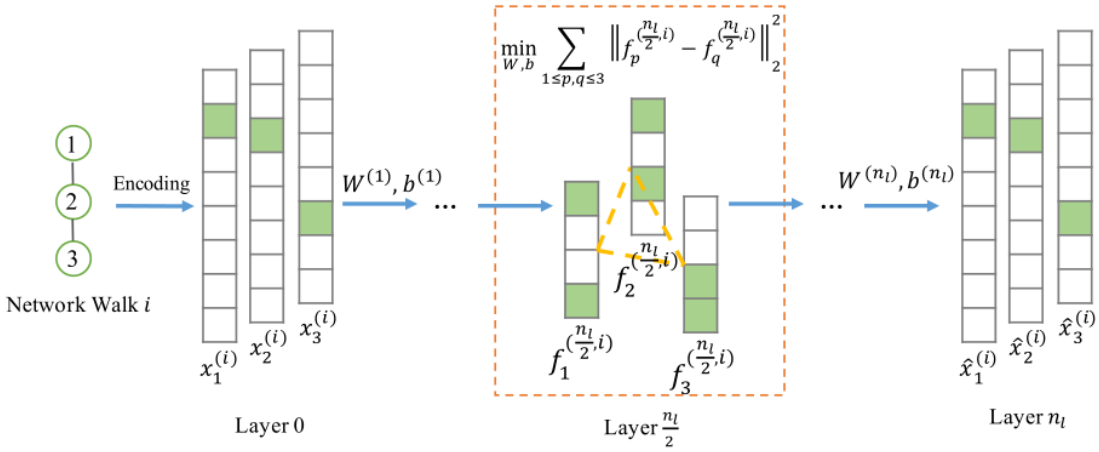

图 2: 长度为 3 的网络行走的团嵌入示意图

该算法利用一个深度自编码器神经网络最小化每次游走中节点之间的成对距离，使用自编码器的重构损失作为正则化项。自编码器的输入和输出是 one-hot 编码的向量，也就是，对于给定的节点 $x_p^{(i)} \in \mathbb{R}^n$，n 个元素中只有 1 个是 1，其他的都是 0，1 的位置为节点对应的索引。NetWalk 的目标是学习每个输入随机游走 $\{x_p^{(i)}\}_{i=1}^{l}$ 的潜在表示。其中 l 是游走长度，$\mathbf{W}^{(l)}$ 是权重矩阵，$b^{(l)}$ 为偏置向量，$f^{(l)}$ 为每一层的输出。形式上，给定 one-hot 编码网络游走 $\{x_p^{(i)}\}_{i=1}^{l}$，NetWalk 学习 n_l 层自编码器网络中的低维表示：

$$f^{(\frac{n_l}{2})}(x_p^{(i)}) = \sigma(\boldsymbol{W}^{(\frac{n_l}{2})} h^{(\frac{n_l}{2})}(x_p^{(i)}) + b^{(\frac{n_l}{2})}) \tag{3.1}$$

$$h^{(\frac{n_l}{2})}(x_p^{(i)}) = \sigma(\boldsymbol{W}^{(\frac{n_l}{2}-1)} h^{(\frac{n_l}{2}-1)}(x_p^{(i)}) + b^{(\frac{n_l}{2}-1)}) \tag{3.2}$$

这里，$\sigma(z) = 1/(1+\exp(z))$ 是 sigmoid 函数；$n_l \geq 2$；$f^{(0)}(x_p^{(i)}) = x_p^{(i)}$。在自编码器网络中，输出 $f^{n_l}(x_p^{(i)})$ 等于 $x_p^{(i)}$ 。因此，如果使用 $l2$ 范数最小化重构误差，目标函数为:

$$J_{AE} = \frac{1}{2} \sum_{i=1}^{|\Omega|} \sum_{p=1}^{l} \left\| f^{(n_i)}\left(x_p^{(i)}\right) - x_p^{(i)} \right\|_2^2 \tag{3.3}$$

NetWalk 最小化第 $n_l/2$ 层嵌入空间中每个随机游走的所有节点之间的成对距离，可以形式化描述为:

$$J_{\text{Clique}} = \sum_{i=1}^{|\Omega|} \sum_{1 \leqslant p,q \leqslant l} \left\| f^{(\frac{n_l}{2})}\left(x_p^{(i)}\right) - f^{(\frac{n_l}{2})}\left(x_q^{(i)}\right) \right\|_2^2 \tag{3.4}$$

由于输入输出向量的稀疏性，NetWalk 考虑了一个具有稀疏性参数 ρ 的稀疏自编码器，并使用 Kullback-Leibler 散度 [48] 来惩罚:

$$KL\left(\rho \| \hat{\boldsymbol{\rho}}^{(l)}\right) = \sum_{j=1}^{d} KL\left(\rho \| \hat{\boldsymbol{\rho}}_j^{(l)}\right) = \sum_{j=1}^{d} \rho \log \frac{\rho}{\hat{\boldsymbol{\rho}}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\boldsymbol{\rho}}_j} \tag{3.5}$$

其中 $\hat{\rho}^{(l)} = \sum_{i=1}^{|\Omega|} \sum_{p=1}^{1} f^{(l)}(x_p^{(l)})/(l \times |\Omega|)$ 是隐藏层中神经元的平均激活量。这种稀疏约束惩 $\rho^{(l)}$ 与 ρ 的偏差。定义总体代价函数为:

$$J(\boldsymbol{W}, b) = \sum_{i=1}^{|\Omega|} \sum_{1 \leqslant p,q \leqslant l} \left\| f^{\left(\frac{n_1}{2}\right)}\left(x_p^{(i)}\right) - f^{\left(\frac{n_l}{2}\right)}\left(x_q^{(i)}\right) \right\|_2^2$$

$$+\frac{\gamma}{2}\sum_{i=1}^{|\Omega|}\sum_{p=1}^{l}\left\|f^{(n_l)}\left(x_p^{(i)}\right)-x_p^{(i)}\right\|_2^2 \tag{3.6}$$

$$+\beta\sum_{l=1}^{n_l-1}\sum_{j}KL\left(\rho\|\hat{\boldsymbol{\rho}}_j^{(l)}\right)+\frac{\lambda}{2}\sum_{l=1}^{n_l}\left\|\boldsymbol{W}^{(l)}\right\|_F^2$$

其中 $|\Omega|$ 为随机游走数，l 为随机游走长度。权值衰减项减小了权值的大小，并有助于防止过拟合。γ，β 和 λ 控制相应惩罚项的权重。

3.2.3 边编码

NetWalk 学习节点的低维向量表示，这使得能够基于聚类检测节点异常。此外，NetWalk 还可以检测异常边。为了确定一边是否异常，NetWalk 构建一个查找表，根据所学的图的表示实时编码新的边。对于无向网络。这个算子必须是对称的。也就是说，对于任何边 (u,v) 或 (v,u)，边的表示应该是相同的。NetWalk 使用了在边编码中表现出良好性能的 Hadamard 算子。假设算法 2.1 学习到的节点 v 的 d 维表示为 $f(v)$，则每条边 (v,u) 在 Hadamard 算子下的表示为 $[f(v)\cdot f(u)]_i = f_i(v)\times f_i(u)$。

3.3 增量维护网络表示

为了应对动态图的快速变化特性，NetWalk 更新图的表示，而不必明确地维护图的完整细节。每一个添加或删除的边都会影响一些随机游走，这些游走将被用来更新当前的节点表示。NetWalk 设计了一种基于蓄水池的算法来维护一个紧凑的记录，该记录由每个节点的一组“邻居”组成，并基于每个节点的更新游走。

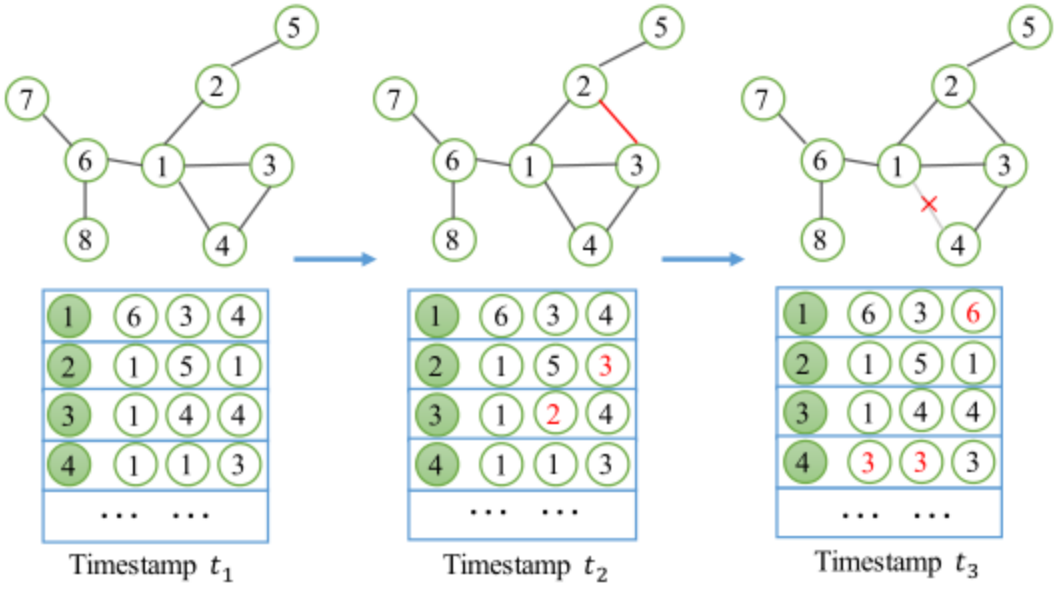

图 3: NetWalk 的节点蓄水池示意图

NetWalk 的节点蓄水池定义如下：对于每个节点 $v\in\mathcal{V}$，对应的节点库 S_v 是一个带有 ψ 项的节点集合，节点集合里的元素通过从 v 的邻居 $ne_v=\{u\mid(u,v)\in E,u\neq v\}$ 进行采样。

给定一个边流，NetWalk 为每个节点 v 维护一个大小为 ψ 的存储池，存储池中的每个项都从 v 的邻居中随机得到。因此，当新的边到达时，需要更新存储池。对于每条新增边 (u,v)，更新规则如下:

(1) 更新节点 u 和 v 的度数:$D_{u,u}=D_{u,u}+1$，$D_{v,v}=D_{v,v}+1$；

(2) 对于蓄水池 S_u 中的每个项，用概率 $D_{u,u}$ 将旧项替换为新项，用概率 $1-1/D_{u,u}$，保留原项；

(3) 对于蓄水池 S_v 中的每一项，以概率 $1/D_{v,v}$ 将旧项替换为新项，用概率 $1-1/D_{v,v}$ 保留原项。

在删除边的情况下，以类似于前面提到的规则选择存储。在这种情况下，需要首先更新度矩阵，然后用对应节点的剩余邻居替换被删除的项。

如图 3 所示，当在时间戳 t_2 处添加 (v_2, v_3) 时，v_2 和 v_3 的相应储水池添加 v_3 的概率为 1/3，添加 v_2 的概率为 1/3。类似地，当 (v_1, v_4) 在时间戳 t3 被删除时，用 v_3 替换被删除的项 v_1，其概率为 1(对应节点 v_4 只剩下一个邻居)，用 v_3 或 v_6 替换被删除的项 v4 的概率为 1/2。

当边 (u, v) 到达时，更新相应节点的存储后，随机游走也应相应的更新。对每条新添加的边 (u, v), 需添加的游走集合定义为 $\Omega_+ = \{(u_1, u_2, \ldots, u_i, u, v, v_1, v_2, \ldots, v_j) \vee (v_1, v_2, \ldots, v_i, v, u, u_1, u_2, \ldots, u_j) \mid i+j = l-2\}$，它是长度为 l 的随机游走集合，其中包含新边 (u, v)。每个连通节点对 (u_m, u_n) 的转移概率为 $p(u_m, u_n) = 1/D_{u_m,u_n}$。对于每条需要移除的边 (u', v')，游走定义为 $\Omega_- = \{\omega \mid \forall\omega \in \Omega \wedge ((u', v') \in \omega \vee (v', u') \in \omega)\}$。然后，NetWalk 将继续训练模型。

3.4 异常检测

通过 NetWalk 学习到的低维表示可以用于许多下游应用，如链路预测、异常检测和社区检测。NetWalk 定义动态网络中的异常检测问题如下: 给定节点表示 $f_{\mathbf{W},b}(x_p^{(i)}) \in \mathbb{R}^d$ 或相应边的低维表示，将现有的所有低维表示划分出 k 个簇，对于任何新到达的节点或边，不属于任何现有簇的可能包括以下场景：节点或边是一个异常；节点或边标志着一个新簇的开始。这两种情况很难区分，除非随后接收到更多的流数据。在 NetWalk 中，找到离每个点最近的簇。NetWalk 使用欧几里德距离作为相似性度量，由 $\|c - f(\cdot)\|_2$ 给出，其中 c 是聚类中心，$f(\cdot)$ 是每个节点或边的低维表示。每个数据点的异常得分设为其到任何聚类中心最近的距离。当新的边进来时，需要相应地更新聚类中心。NetWalk 利用流式 k-means 聚类 [32]。在计算吸收新点后的新聚类中心时引入一个衰减因子 α。使用参数 α 来控制现有簇中“旧”数据点的重要性。

假设已存在的有 n_0 个数据点 $\{x_i\}_{i=1}^{n_0}$ 和 n' 个新数据点 $\{x'_i\}_{i=1}^{n'}$，在时间戳 T' 被该簇吸收时，质心 c 可以按照以下方式更新：

$$c = \frac{\alpha c_0 n_0 + (1-\alpha)\sum_{i=1}^{n} x'_i}{\alpha n_0 + (1-\alpha)n'} \tag{3.7}$$

其中 c_0 是以前的簇中心。衰减因子 α 被选为 0.5，用于忽略较老的实例，这类似于指数加权移动平均。

4 复现细节

4.1 与已有开源代码对比

本次复现的论文代码部分采用公开的相关源代码，包括团嵌入网络学习表示过程和网络动态更新过程两部分代码，整体模型框架与论文思路相同。在具体实现的过程中，对于不同类别的数据集需要做不同的数据预处理，调整网络学习过程中的相关参数，以及网络动态更新过程中的边数及时间戳个数，使模型效果达到最优。算法 1 为 Netwalk 团嵌入表示学习过程，算法 2 为 Netwalk 动态更新过程，伪代码描述如下：

Procedure 1 Clique Embedding of NetWalk

Input: Network walk set Ω .
Output: Network representations $f^{n_l/2}(\mathrm{x}_p^{(i)})$.
Set latent dimension d, sparsity ρ, weight control parameters γ,β and λ.
Randomly initialize $\{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{n_l}$.
Construct input vector $\mathrm{x}_p^{(i)} \in \mathbb{R}^n$ for vertex p in walk i, $1 \leqslant p \leqslant l, 1 \leqslant i \leqslant |\Omega|$.
while *not stopping criterion* **do**

    Perform a feedforward pass to compute $f^{(\ell)}(\mathbf{x}_p^{(i)})$.

    Compute error.

    Backpropagation and update autoencoder parameters according to errors.

end
Compute embedding results $f^{n_l/2}(\mathrm{x}_p^{(i)})$.

Procedure 2 Network Representation Maintenance

Input: Network walk set Ω ; a streaming edge set $E^{(t)}$; saved clique embedding model.
Output: The updated Ω , the updated embedding clique model.
// dynamic walks generation
for (u, v) *in the streaming edge set* $E^{(t)}$ **do**

    Update vertex set V.

    Update degree matrix D.

    Update the reservoirs S_u and S_v using the rules described in Section 3.3.

    Generate the network walk sets Ω_+ for new edges and **Ω−** for deleted edges, respectively.

end
// model update
Load the saved embedding model.
Train the model with the dynamic network walk set Ω_+ with a small sample set of walks from Ω, or with the updated walk set $\Omega - \Omega_-$ if with edge deletion.
Update network representations $f^{n_l/2}(\mathrm{x}_p^{(i)})$.
Save the updated clique embedding model.

4.2 实验环境搭建

本次复现使用 python 作为编程语言，主要用到的包名及版本信息如下：

python==3.6

tqdm==4.64.1

scipy==1.5.4

numpy==1.19.5

networkx==2.5.1

tensorflow==2.6.2

matplotlib==3.3.4

scikit-learn==0.24.2

4.3 界面分析与使用说明

使用 Netwalk_update.py 预处理图数据，从初始图中生成相应随机游走的 one-hot 编码，并实现节点蓄水池的初始化和动态更新，动态维护图结构；模型整体框架为 Model.py 文件，是一个基于团嵌入的深度自编码器，使用 Netwalk.py 文件调用深度自编码器网络模型，训练学习节点的团嵌入表示，将学习到的节点维度表示保存到相应文档，并生成显示出相应的效果图；使用 Netwalk_anomaly.py 文件

动态更新网络节点和边的编码表示，对顶点和边流进行异常检测，输出每个时间戳的异常评分以及预测准确度。

5 实验结果分析

5.1 数据集描述

为了验证所提出的 NetWalk 模型的性能，在不同领域的各种动态网络上进行了实验，如表 1 所示。UCI 信息网络是基于加州大学欧文分校学生的在线社区。每个顶点代表一个用户，边代表一个消息交互。Digg5 是新闻聚合网站 digg.com 的回复网络。每个节点是一个网站用户，每条边表示两个用户之间的回复。arXiv hep-th 数据是一个来自高能物理理论类别 (hep-th) 的协作网络。每个节点代表一个作者，边代表协作。DBLP 数据也是 DBLP 计算机科学书目作者的合作网络。Email-Eu-Core 时态网络数据集，该网络是根据欧洲某大型研究机构内部成员的电子邮件往来关系所构建的。Cora 数据集由机器学习论文组成，是近年来图深度学习很喜欢使用的数据集。各数据集具体描述如表 1 所示。

表 1 各数据集描述

Datasets	Vertex	Edge
UCI Messages	1899	59835
arXiv hep-th	23127	2673133
Digg	30398	87627
DBLP	300647	807700
Email-Eu-Core	1005	25571
Cora	2708	5429

5.2 Netwalk 团嵌入表示学习结果

可视化。在网络表示学习阶段，NetWalk 以初始网络作为输入，学习每个顶点的潜在表示。在这里，我们通过将我们的方法应用于 Zachary 的空手道网络，展示了 NetWalk 中派系嵌入的编码能力。图 3 左半部分显示了原始网络，其中顶点颜色表示每个顶点的所属社区。首先将图网络中的 50% 边流传入模型中学习初始节点表示，其次将剩余边流分几个时间戳依次传入，并动态更新节点表示。将原始图结构和学习到的维度表示进行对比，图 4 左半部分为原始网络的图结构，图 4 右半部分展示了 NetWalk 学习到的二维表示。值得注意的是，可以在我们的方法学习到的向量表示空间中找到线性可分簇。

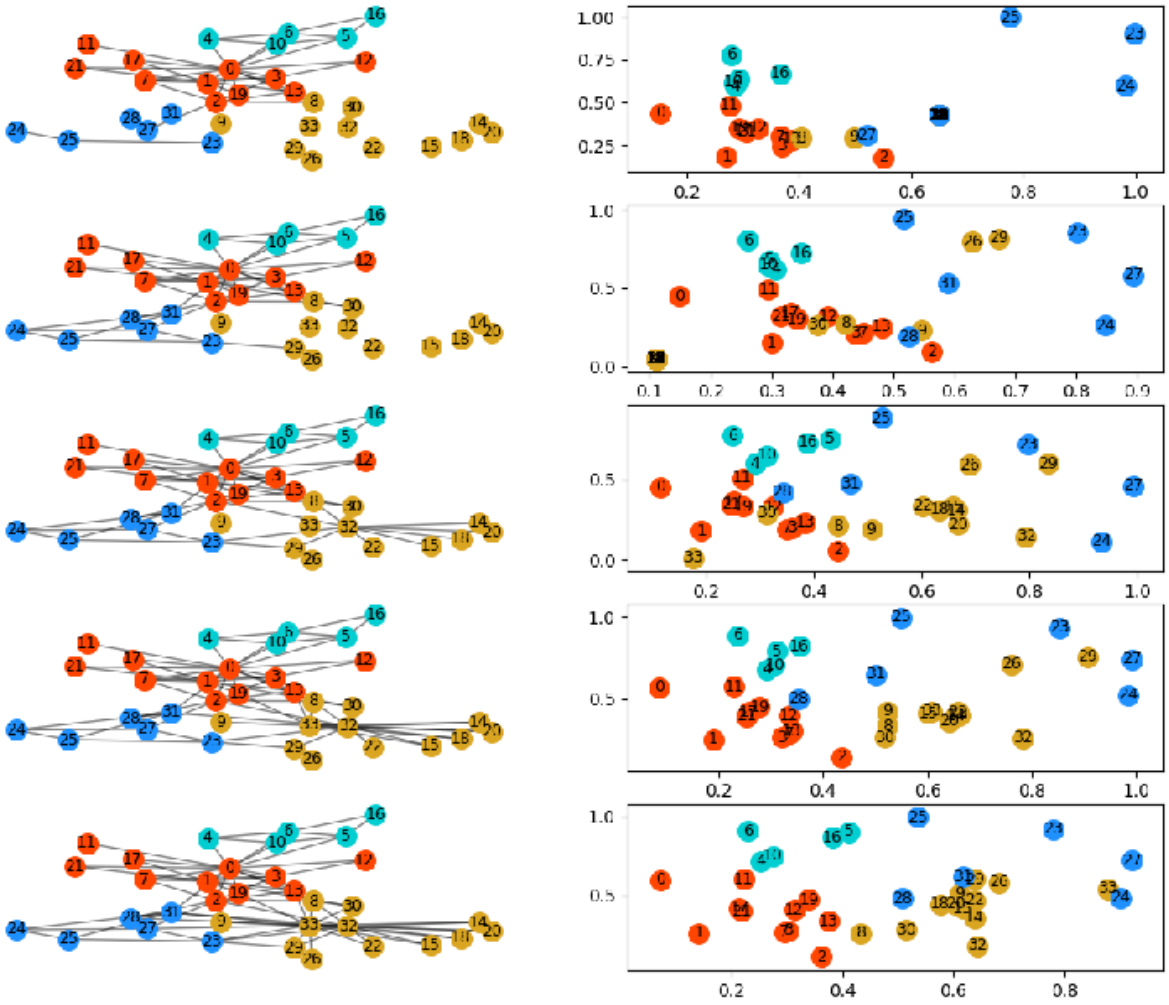

图 4: 在 Zachary 的空手道网络中嵌入结果

5.3 异常识别结果

在两种设置中评估 NetWalk: 静态设置和流设置。在静态设置中，网络的前 50% 边用于训练，其余的传入边用于测试。顶点表示是脱机学习的。然后，我们使用这些表示对训练边进行编码和聚类。测试边缘根据它们到最近的聚类中心的距离进行评分和排名。目的是量化 NetWalk 的网络表示在异常检测任务中的有效性。由于地面真实异常数据采集的挑战，我们使用异常注入方法创建异常边缘，分别注入 1%，5%，10% 的异常来检测模型效果。曲线下面积 (area under curve, AUC) 评分用于衡量所有方法的预测能力。我们通过计算从训练边生成的聚类中到最近中心的距离来对所有编码的测试边进行排名和评分，如表 2 所示。

表 2 静态设置下异常检测结果

Datasets	1%	5%	10%
UCI Messages	0.73	0.71	0.69
hep-th_sub	0.69	0.68	0.63
Digg_sub	0.72	0.74	0.70
DBLP_sub	0.71	0.68	0.67
Email-Eu-Core	0.76	0.73	0.69
Cora	0.75	0.74	0.72

在流设置中，我们再次使用前 50% 的边进行训练，以构建初始网络表示和集群。测试边依次到达

并在线处理。换句话说，所有测试边在任何给定时间都只能部分可见。为了便于比较，我们将流边分割为几个快照。根据不同数据集的测试集大小，每个快照的边数按数据集大小设置。对于每个到达的快照，NetWalk 更新相应的网络表示、集群和异常分数。动态网络异常检测精度如下图 5 所示。

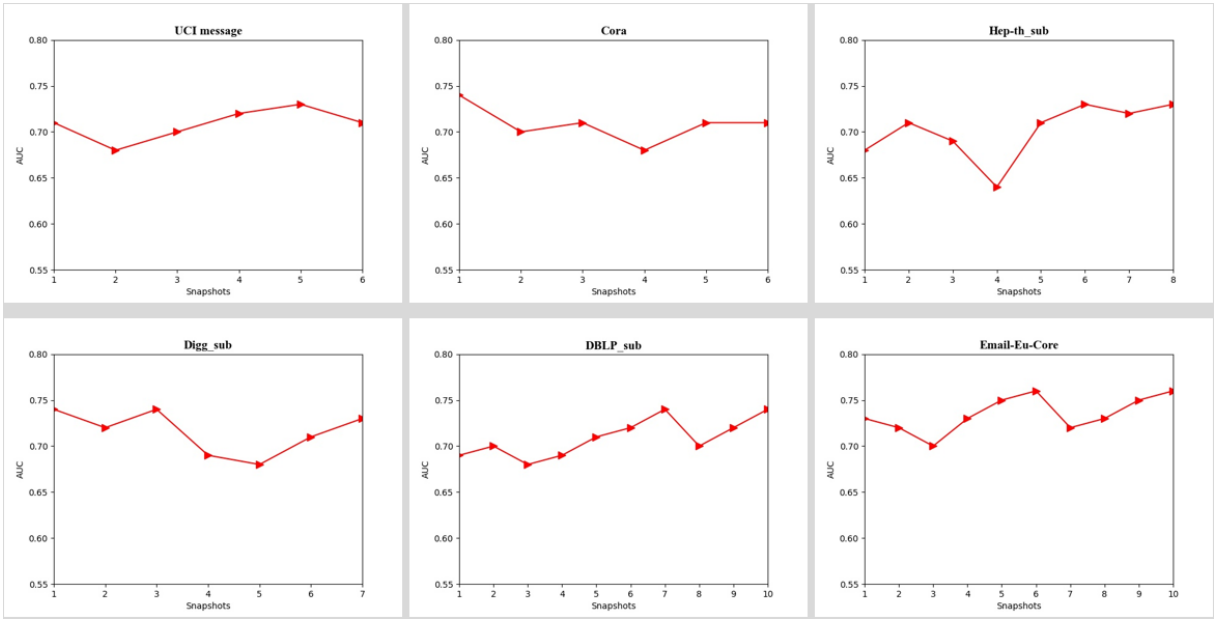

图 5: 动态网络异常检测精度（5% 异常注入）

根据实验结果可能看到，Netwalk 可以非常好的学习到节点以及边的编码嵌入，能使得结构相似的节点具有相似的表示；在异常检测方面，无论是在静态设置还是动态流设置的情况下，Netwalk 算法都具有较好的效果。

5.4 消融实验

NetWalk 框架涉及许多可能影响其性能的参数。检查两个异常检测任务 (UCI Messages 和 Digg) 的性能变化。我们改变每个顶点的样本数量 (ψ)，顶点表示的维数 (d)，训练数据百分比和步行长度 (l)，以确定它们对异常检测的影响。除正在测试的参数外，其他参数均采用默认值。结果如下图 6 和 7 所示。

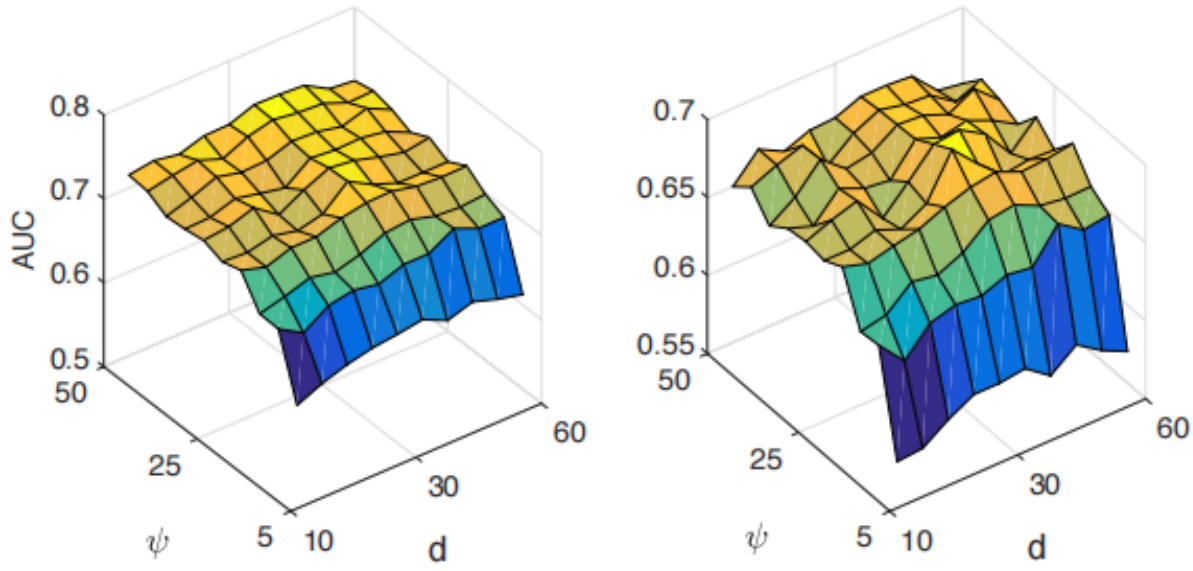

图 6: 不同参数对对 UCI Messages(左) 和 Digg(右) 异常检测 AUC

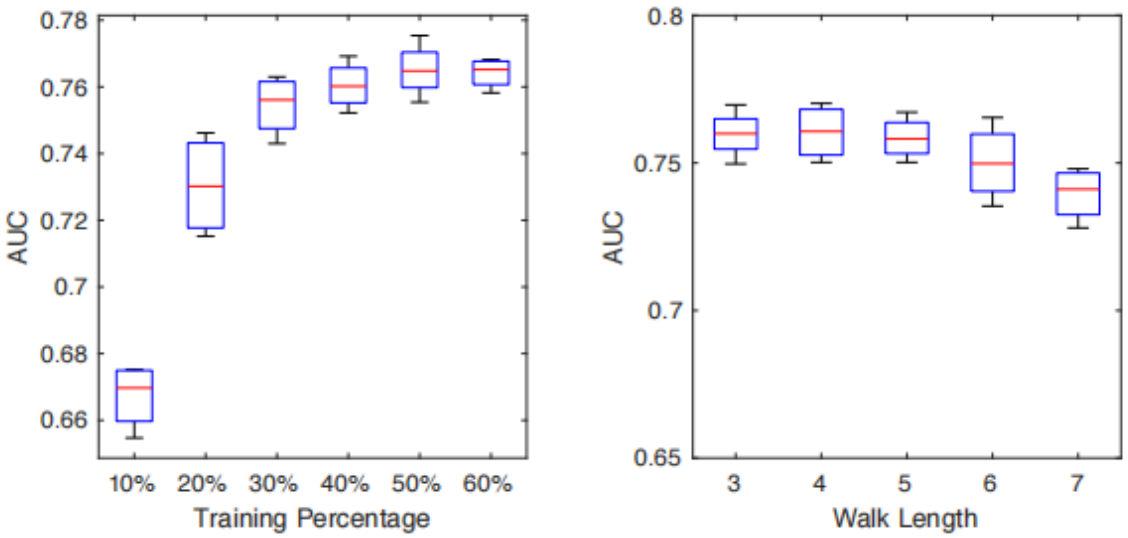

图 7: 初始网络训练百分比的稳定性 (左)，UCI 消息上 5% 异常的网络行走长度 (右)

6 总结与展望

NetWalk 来检测动态网络中的异常，通过学习忠实的网络表示，可以随着网络随着时间的推移而动态更新。首先通过使用从初始网络中提取的大量网络行走来学习潜在的网络表示。这些表示是通过派系嵌入获得的，它联合最小化每个网络行走的顶点表示的成对距离，以及作为全局正则化的自动编码器重建误差。基于低维顶点表示，采用聚类技术对网络异常进行增量动态检测。利用四个读世界数据集对异常检测任务进行了定量验证，结果表明，NetWalk 算法计算效率高，在异常检测方面具有较好效果。

参考文献

[1] AGGARWAL C C, ZHAO Y, YU P S. Outlier detection in graph streams[C]//Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany. 2011.

[2] RANSHOUS S, HARENBERG S, SHARMA K, et al. A Scalable Approach for Outlier Detection in Edge Streams Using Sketch-based Approximations[C]//SDM. 2016.

[3] MANZOOR E, MILAJERDI S M, AKOGLU L. Fast Memory-Efficient Anomaly Detection in Streaming Heterogeneous Graphs[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016: 1035-1044.

[4] AGGARWAL C C, ZHAO Y, YU P S. On Clustering Graph Streams[M]//Proceedings of the 2010 SIAM International Conference on Data Mining (SDM): 478-489.

[5] ESWARAN D, FALOUTSOS C. SedanSpot: Detecting Anomalies in Edge Streams[J]. 2018 IEEE International Conference on Data Mining (ICDM), 2018: 953-958.

[6] PEROZZI B, AL-RFOU R, SKIENA S. DeepWalk: Online Learning of Social Representations[C/OL]//. New York, NY, USA: Association for Computing Machinery, 2014: 701-710. https://doi.org/10.11

45/2623330.2623732. DOI: 10.1145/2623330.2623732.

[7] TANG J, QU M, WANG M, et al. LINE: Large-scale Information Network Embedding[J]. Proceedings of the 24th International Conference on World Wide Web, 2015.

[8] GROVER A, LESKOVEC J. node2vec: Scalable Feature Learning for Networks[J]. KDD : proceedings. International Conference on Knowledge Discovery & Data Mining, 2016, 2016: 855-864. DOI: 10 .1145/2939672.2939754.

[9] RIBEIRO L F R, SAVERESE P H P, FIGUEIREDO D R. struc2vec: Learning Node Representations from Structural Identity[J]. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.

[10] MAHDAVI S, KHOSHRAFTAR S, AN A. dynnode2vec: Scalable Dynamic Network Embedding[J]. 2018 IEEE International Conference on Big Data (Big Data), 2018: 3762-3765.

[11] ZHENG L, LI Z, LI J, et al. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN[C]//International Joint Conference on Artificial Intelligence. 2019.