基于图表征学习的电信网络拓扑还原

1 介绍

网络拓扑是服务工程师最基本、最广泛的网络信息依据，准确的、完整的、实时度高的业务路径拓扑还原，在运维领域非常有竞争力。拓扑模块上叠加告警、性能、RCA 等叠加信息，可以快速实现故障的诊断。拓扑也是网络可视化最基本的元素。但是目前的拓扑不完整在各个局点和场景中普遍存在。此合作项目的主要目标是利用图神经网络技术从结构化和非结构化的故障相关信息（如告警、日志等）等还原设备的拓扑关系，实现辅助根因告警的定位以及拓扑的可视化，以满足运维的需要。本报告作为第一阶段的输出主要包含三个内容。拓扑还原旨在利用已有的告警以及拓扑信息对缺失拓扑进行还原，问题定义和链接预测基本一致。因此我们首先对链路预测问题的相应方法进行了调研（Section 2）。进一步我们对华为的数据进行相应的分析，并进行了相关的实验（Section 3), 实验探索发现，对于链路预测问题，基于子图编码的方式较其他的图表征学习方法有十分明显的优势，在华为数据集上利用子图编码以及相应的设备以及告警信息，可以实现较好的预测效果（AUC 大于 80%）。图神经网络（GCNs）在学习图的表示方面已经引起了广泛的关注，应用也非常广泛。图神经网络的灵感主要来自最近的一些深度学习方法，因此可能会继承不必要的复杂度和冗余计算，计算的复杂度也影响了其在工业系统中的应用，为了更好的适用大规模图学习问题，我们提出了一种基于图粗化的高效图神经网络训练方式并给出了相关的理论证明（Section 4）。最后我们给出了相应的总结 (Section 6)。

2 链接预测

图结构常被用来存储和表示“多对多”关系的数据，在复杂网络中应用地尤为广泛。比如在社交网络中，用图来存储用户之间的好友关系；在生物学中，用图来展示不同蛋白质之间的相互作用；在知识图谱中，用图来揭示知识领域的动态发展过程（Knowledge GraphEmbedding）。一张图由节点（Vertex）和边（Edge）组成，其中点用来表示对象，边用来表示对象之间的关系，用数学化的形式表示为图 $G = (V, E)$，其中 V 表示节点的集合，$E \subseteq |V| \times |V|$ 表示节点之间所形成的边的集合。

然而由于已知的对象间关系往往不够完整；即便在某一时间得到完整的关系，关系也可能会随着时间的发展而产生变化。因此在理论研究与实际应用中，一个很重要的问题就是：一张图中哪些节点之间可能存在被遗漏掉的边? 这就是链接预测要解决的问题。链接预测接受一张图作为输入，经过对图中信息的处理加工，可以用来预测任意两个点之间存在链接的可能性有多大。

链接预测的研究在实际生产生活中有着重大的作用。例如：帮助运营商发现社交软件用户之间的潜在好友关系；在学术引文网络中预测作者之间的合作关系；预测蛋白质之间的相互作用等等。链接预

测的主要目的是揭示网络背后的一些深层次的关系，从而预测网络中缺失的边以及未来可能存在的边。

在如此广泛的应用前景下，近年来学者们对链接预测的方法进行了丰富的研究。在本文中，我们主要把链接预测的方法分为三类，分别是：(1) 基于启发式方法的链接预测；(2) 基于图嵌入的链接预测；(3) 基于图神经网络的链接预测。其中，

- 启发式方法最为传统，是用某种评分机制判断所有点之间可能存在链接的概率，然后取一个阈值作为有链接与无链接的区分。这种方法思路简单，并且也可以得到较为不错的结果。
- 基于图嵌入的方法则是先将图数据进行降维处理，用向量来表示图中所有的点，然后把这些向量作为链接预测的输入，进而用传统的二分类方法判断链接是否存在。
- 基于图神经网络的方法是先提取出每个点周围处的一个封闭子图，然后以此作为一个图神经网络 (GNN) 的输入，让 GNN 自动地从图的拓扑结构中学习出预测链接的方法。

本章将简单介绍这三种链接预测方法。前文中我们已经介绍了相关背景以及链接预测的意义，并引出了三类链接预测方法；接下来，我们将在第 1 节给出链接预测问题的一个规范的定义；在第 2∼4 小节，分别介绍三种方法的核心思想以及主流算法；在第 5 节，我们综合地对三类方法进行一个比较。

2.1 问题定义

2.1.1 图的定义

现在，让我们考虑在一张给定的图上进行链接预测。用 $G = (V, E)$ 来表示这张图，其中 V 表示图中所有的节点 (vertex), E 表示图中观测到的边 (edges)。链接预测问题中的图基本上都并非完全连通图，所以 $E \subset |V| \times |V|$。有时，图上的边是有方向的，例如在社交网络中 A 用户关注了 B 用户，而 B 用户不一定关注 A 用户，这种网络对应的边也是有方向的，故称这种图为**有向图 (directed graph)**；与之相对的则是**无向图 (undirected graph)**。

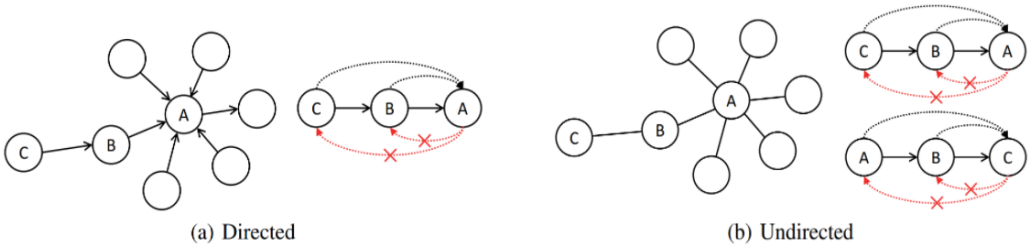


Figure 1: 有向图和无向图.

对于 $x, y \in V$, 我们用 $\Gamma(x)$ 来表示所有与 x 相邻的点，也即是 x 的一阶邻居，用 $d(x, y)$ 来表示点 x, y 之间最短路径的长度。将所有的点以某种顺序编号后，v_i 可用来表示图中的点，如果在点 v_i, v_j 之间存在一条链接，可用 $(v_i, v_j) \in E$ 来表示。

2.1.2 邻接矩阵

常见的描述图中的边的方法有两种，一是列出所有的边 $E = \{(v_i, v_j)\}$，二是使用邻接矩阵 (adjacency matrix) 来表示点之间的连接情况。邻接矩阵定义为

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

显然，在无向图中 A 是一个实对称阵，在有向图中 A 则不一定对称。邻接矩阵具有很多丰富的性质，比如，A_{ij} 处的值为点 v_i, v_j 之间存在的边的个数，而 A^2 中 (i,j) 位置的值是点 v_i, v_j 之间存在的长度为 2 的路径的个数。其证明非常简单：

$$A^2_{ij} = \sum_k^N A_{ik} A_{kj} \tag{2}$$

求和式里面的值只有在 $A_{ik}, A_{kj} \neq 0$ 的时候才不为 0，亦即说明这是一条位于 (v_i, v_j) 之间的路径。由此类推，A^m_{ij} 所表示的正是点 v_i, v_j 之间，长度为 m 的路径的个数。

2.1.3 封闭子图

由于在链接预测问题中，我们经常需要考虑节点对 (x, y) 周围的节点形成了怎样的拓扑结构，并以此判断链接的存在，所以我们还需要引入封闭子图 (enclosing subgraph) 的概念。点 x, y 的一个 h 阶封闭子图定义为：

$$G^h_{x,y} = \{i | d(i,x) \leqslant h \text{ or } d(i,y) \leqslant h\} \tag{3}$$

它包含了所有与点 x, y 之间距离小于等于 h 的点。

2.1.4 评估方法

链接预测常常用机器学习问题中典型的指标：受试者工作特征曲线（ROC）下的面积（AUC）来评估。几乎所有链接预测模型都会对任意两个点之间打分，如果点的值超过阈值则认定其存在链接，反之则没有；实际实验中，我们先在保存一张图连通性的基础上随机地删去一些边，然后进行链接预测模型的训练。训练后，我们随机地选取一些不存在链接的节点对 (v_1, u_1)，并随机地选取一些被我们删去链接的节点对 (v_2, u_2)，比较两者在链接预测模型下得到的分数。若进行了 n 次比较中，有 n_1 次 (v_2, u_2) 对得到的分数更高，n_2 次得到的分数相等，那么可以直接用以下公式进行计算 AUC [29]：

$$AUC = \frac{n_1 + 0.5 n_2}{n} \tag{4}$$

2.2 启发式链接预测

启发式方法是比较传统的一类链接预测方法，在深度学习算法成熟之前，被广泛地应用于链接预测。之所以称这类方法为“启发式” (heuristic), 是因为在不同的图结构下，首先需要“启发”地判断什

么样的点之间更可能存在链接（比如拥有较多相同邻居的点更容易存在链接），然后由此设置一种评分算法，给所有的点对 (v_i, v_j) 之间进行打分。两点之间的分数，就是它们之间存在链接的概率。我们可以设置一个阈值，认为所有概率超过此阈值的点对 (v_i, v_j) 之间存在链接，正如二分类问题常见的处理方法。

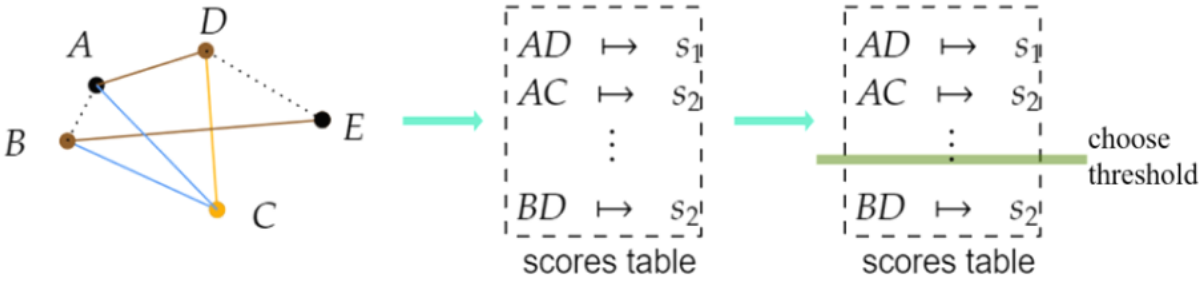


Figure 2: 启发式链接预测框架.

启发式方法的核心就在于选取评分函数。在评估两个点之间存在链接的概率时，可以依据所考察点的范围对启发式方法进行分类。一类是只考察图的局部结构，我们称之为局部式启发；另一类是通过一些方法近似地从整体考察图的结构，我们称之为全局式启发。

2.2.1 局部式启发

局部式启发虽然都是只用图中局部的信息来判断链接存在的可能性，但是根据判断过程中设计的封闭子图的阶数，还可进一步将局部式启发分为 1 阶局部式启发以及 2 阶局部式启发等。

一阶局部式启发

- **Common Neighbors(CN)**
  CN 是最简单、同时也应用最广泛的一种启发式方法。对于一对点 $x, y \in V$, Common Neighbors 的评估函数为

$$S(x, y) = |\Gamma(x) \cap \Gamma(y)| \tag{5}$$

  这种假设很直观：如果两个点具有更多的相同邻居，那么这两个点之间的关系可能会很密切，它们之间将更有可能存在链接。Common Neighbors 只用到了两个节点的一阶封闭子图，因此对应的是一阶局部式启发方法。虽然简单，但是 Common Neighbors 的效果却很好，在 Zhou 和 Lu 所进行的实验中，Common Neighbors 在九种局部式启发方法中有着最好的平均效果 [29]。在日常生活中，我们熟悉的腾讯 QQ 进行好友推荐的方法也正是 Common Neighbors。

  类似于 CN，还有很多种常见的一阶局部式启发方法：

- **Jaccard Coefficients**
  Jaccard 系数只在 CN 的基础上做了细微的变化：

$$S(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{6}$$

其分子部分就是 CN 的值，而分母部分变成了 x, y 点一阶封闭子图中点的总数。从定义中理解，Jaccard Coefficients 的评分等于随机从 x, y 的一阶封闭子图中选择一个点，这个点恰好同时与 x, y 相邻的概率。它显然也是一阶的局部启发。Jaccard Coefficients 方法相对于 CN 的好处在于，它对每两个点之间的分值限制在了 0 至 1 内，避免了图中有些节点相邻十分众多的情况。在那种情况下，图中点的联系比较密集的地方会被 CN 方法评以很高的分数，从而造成链接预测过程中对节点密集区域的偏好。

- **Salton Index**
  类似于 Jaccard Coefficients，在 CN 的基础上进行归一化调整，除以两节点度的几何平均数。

$$S(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{k(x) \times k(y)}} \tag{7}$$

- **Preferential Attachment**

$$S(x,y) = k_x \cdot k_y \tag{8}$$

Preferential Attachment 考虑了节点的邻居对于其之间形成链接的影响，若节点在社区中的影响力非常大，则它与其它节点之间形成链接的可能性也越大。

二阶局部式启发

- **Adamic Index**

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k_z} \tag{9}$$

对两节点的共同邻居的度数的对数取倒数，这不仅衡量了它们之间的共同性，还减少了周围连接度很高的点对它们的影响.

- **Resource Allocation**

$$S(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k_z} \tag{10}$$

RA 是对两节点的共同邻居的度数取倒数并相加，与 Adamic Index 大抵相同。

2.2.2 全局式启发

全局式启发使用整张图的拓扑结构对每两个点之间进行打分，因此保存更多的信息，但计算起来也更费时间。比较经典的方法有：

- **Katz index**

$$S(x,y)=\sum_{l=1}^{\infty}\beta^l|walks^{<l>}(x,y)|=\sum_{l=1}^{\infty}\beta^l[A^l]_{x,y} \tag{11}$$

基于前文的描述我们知道，在一个图中，对所有可能的长度为 n 的路径求和，等于邻接矩阵 A 的 n 次方的元素值。这里有 $0<\beta<1$，因此随着 l 的增加，高阶项变得不那么重要。主要保存的还是与两个点距离较近路径的信息。

- **Shortest Path**

$$S(x,y)=-|d(x,y)| \tag{12}$$

其中 $d(x,y)$ 表示 x,y 间的最短距离。有很多算法帮助我们找到两个点之间的最近距离，比如 Liben-Nowell 算法等。在一张图里，我们认为距离越近的点具有的相似性越高，故式 (12) 可以看作是一种衡量两个点相似度的指标。虽然对于距离较远的节点来说，这种评分方式是有效的，但是在局部，这可能会产生大量相等的评分。相比较之下，Katz index 是一种更全面的衡量标准。

- **Rooted PageRank**

这是来自传统的网络页面打分的算法。该算法假设有一个浏览网页的人有 α 概率随机地跳转到与当前节点相连任意节点，还有 $1-\alpha$ 的概率回到最初的节点。这个过程持续下去，最终会趋于一个平衡的分布。该分布的含义是从起点到任何一个点之间的概率，因此这可以用来衡量两个点之间的关联程度。数学地表示为

$$\pi_x=\alpha P\pi_x+(1-\alpha)e_x \tag{13}$$

$$S(x,y)=[\pi_x]_y+[\pi_y]_x \tag{14}$$

启发式链接预测基于两个点的打分结果进行预测，虽然可以根据不同的图结构调整打分策略，但是由于这种打分只能存在于两两点之间，其泛化效果以及全局的预测效果较弱。随着深度学习的蓬勃发展，人们希望将神经网络强大的学习功能应用在链接预测上。这促进了一类基于图嵌入的链接预测研究。

2.3 基于图嵌入的链接预测

在传统机器学习的诸多任务中，需要的输入往往是欧氏空间的向量，而图数据很难直接表示为这样的向量。最粗暴的方式是采用 one-hot 编码，得到对每个节点的长度为 N 的向量表示，然而这些向量之间相互垂直，并且过于稀疏，直接作为神经网络的输入将无法得到良好的预测结果。因此，我们需要在输入之前对数据进行预处理，而这就是图嵌入的过程。

图嵌入属于表示学习，其目的是将表示图的高维向量空间 V 嵌入到一个低维空间 V' 上 ($\mathrm{dim}V'\ll\mathrm{dim}V$) ，获得图的低维表示。在机器学习的诸多任务中，需要的输入往往是可以传递一定信息的向量，而图结构中，节点与节点的连接关系存储着大量有价值的信息，这种信息不能显式地拿来使用。因此我们以这种信息为度量，寻找可以保存这些信息的低维向量组，就可以作为图中节点们的低维表示了。

图嵌入的作用，不仅在于用低维向量表示图结构，从而减少后期的计算量；还在于获得低维表示的同时，表示向量保留了一定的图结构信息。因此，直接使用图嵌入后两个点的表示向量作为输入，让一个二分类器以此判断链接是否存在的，这就是基于图嵌入的链接预测。

本节将分为 4 个部分，其中第一部分介绍图嵌入的原理与常见框架；第二部分介绍几大主流的图嵌入方法，并在互相之间进行比较；由于优化是图嵌入中非常重要的一环，因此第三部分将主流图嵌入方法中的优化算法单独陈列出来；第四部分介绍使用图嵌入结果进行链接预测的相关方法。

2.3.1 图嵌入链接预测的原理与框架

图嵌入旨在找到图结构的一种表示。最简单的表示图中所有节点的方法，就是使用维数为 $n = |V|$ 的向量，让每一个点 v_i 对应于一个单位向量 $\vec{v}_i = (0, \cdots, 1, \cdots 0)^T$，其中第 i 个元素为 1，其余元素为 0. 这种表示方式叫做 one-hot 编码。

One-hot 编码得到的表示向量是非常稀疏的，维度等于图中所有点的总数，并且每个点与点的表示向量互相垂直，不保存任何信息，必须配合邻接矩阵才能体现图的拓扑结构。如果我们采用一种维度小于 n 的表示，那么 $\vec{v}_i$，$\vec{v}_j$ 之间将不再是互相垂直的，这样的话，表示向量之间的内积、距离以及其它代数运算将可以被赋存储一定信息。如果我们以此作为优化目标，将可以让表示向量保存更多的图中信息。一种很自然的图嵌入的想法即：利用图的拓扑结构。因此一种合理的想法便是：找到原始图结构的某种信息特征，并找到一组可以保存该特征的低维表示。

基于上述思想，大部分图嵌入算法都基于以下框架：

- 确定两点间相似程度的相似函数 $\hat{p}(u, v)$.
- 在低维空间下初始化所有的表示向量 s_u, 选取表示向量之间的评判方法 $p(u, v)$；
- 然后训练模型，使得 $p(u, v) \sim \hat{p}(u, v)$

$\hat{p}$ 和 p 的选择不仅影响表示向量存储信息的方法，也影响向量存储信息的内容。比如若将 $\hat{p}(u, v)$ 设为 $u \cdot v$，意味着内积越大的表示向量将具有更大的相似性。一般的，p 的选择决定了嵌入算法的好坏。Tang 等人在 [19] 提出了一种区分 p 的方法：根据 p 用到的信息范围不同，将相似函数分为一阶相似和二阶相似。

- 一阶相似 $p_1(\cdot, \cdot)$ 只用到两个点彼此的信息，比如这两个点在空间中的距离；
- 二阶相似 $p_2(\cdot, \cdot)$ 用到两个点周围点的信息，比如这两个点的距离与它们其它邻点的相互距离。

2.3.2 几种二阶近似的图嵌入算法

LINE

LINE [19] 的提出并非最早的，然而由于它首先区分了相似函数的一阶和二阶，所以这里我们首先来介绍该算法。LINE 使用相对熵 (KL-divergence) 来衡量 p 和 $\hat{p}$ 的相似程度。KL-divergence 的定义如式 (15)。它衡量的是两个分布之间的差异。由吉布斯不等式可知，$D_{KL}(P \mid\mid Q) \geqslant 0$ 并且当且仅当 $P = Q$ 时该式才等于 0。

$$D_{KL}(P \mid\mid Q) := \sum_i p(x_i) \cdot \ln \frac{p(x_i)}{q(x_i)} \, \mathrm{d}x \tag{15}$$

此外，LINE 简单定义 $\hat{p}(i,j) = \frac{w_{ij}}{W}$，其中 w_{ij} 是点 v_i, v_j 之间的权重，若二者不连通，则权重为 0。省略常数项，最小化 p, $\hat{p}$ 之间的相对熵等价于最小化式 (16)。

$$O = \sum_{i \in V} w_{ij} \log p(v_i, v_j) \tag{16}$$

通过定义不同的相似函数，即可获得一阶相似于二阶相似：

- 一阶相似：定义

$$p_1(v_i, v_j) = \frac{1}{1 + \exp\left(-u_i^T \cdot u_j\right)}$$

- 二阶相似：定义

$$p_2(v_j \mid v_i) = \frac{\exp\left(u_j^T \cdot u_i\right)}{\sum_{k=1}^{|V|} \exp\left(u_k^T \cdot u_i\right)}$$

可以看出，二阶相似要考虑到不仅是两点之间的距离，还有它们相对于图中其它点的距离。因此可以保存更多的图信息。不过计算分母要对空间中所有的点进行内积求和，这在操作上是很难实现的。LINE 采用了 negative sampling 的优化方法，这在第三部分会详细介绍。

DeepWalk

与其同时考虑图中所有的节点以获取二阶相似函数，有一种近似的办法是多次随机地考察图中的一些点。DeepWalk [15] 就是这样一种基于随机游走进行图嵌入的算法。DeepWalk 创造性地将自然语言处理领域的 Skip-Gram 算法扩展到图嵌入问题中，它启发了一类基于随机游走进行图嵌入的研究，同时也是对图嵌入算法研究的先驱之一。DeepWalk 判断图结构中两个节点 u, v 相似度的方法很直接：如果节点 u 和节点 v 在图中的地位就比较接近，则点 u, v 对应的表示向量之间也应该有着较近的距离。它通过在图中每个点附近进行随机游走，构造出一条条由节点组成的链。显然，在大量随机游走的过程中，地位接近的节点 u 和节点 v 会以更高的频率出现在产生的节点链中。我们用式 (17) 来表示在已知节点链 $v_1, v_2, \cdots, v_{i-1}$ 后，下一个节点是 v_i 的概率：

$$\Pr(v_i | (v_1, v_2, \cdots, v_{i-1})) \tag{17}$$

如果我们希望对所有点进行了某种嵌入表示 $v_i \to \Phi(v_i)$ 之后，$\Phi(v_i)$ 可以像 v_i 一样保存图的结构，我们不断地在图上进行随机游走，在游走过程中不断地最大化式 (17) 即可得到想要的表示向量。

$$\max_{\Phi} \Pr(v_i | (\Phi(v_1), \Phi(v_2) \cdots \Phi(v_{i-1}))) \tag{18}$$

然而，对式 (18) 进行优化的复杂度将随着链条长度的增加而快速增加，这在实际工程上是很难做到的。如何进行优化呢?DeepWalk 受自然语言处理中，对单词进行嵌入的一种重要算法 Word2Vec [12] [13] 的启发：

- 用在一串链条中遇到某个节点的概率进行优化，可以等价于用在这个节点周围观测到链条的概率进行优化。即式 (17) 可以转化为式 (19)；

$$\max \Pr((v_{i-1}, \cdots, v_2, v_1)|\Phi(v_i)) \tag{19}$$

- 要同时考虑随机游走过程中遇到此节点之前与之后的两段链条；

$$\max \Pr((v_{i+w}, \cdots v_{i+1}, v_{i-1}, \cdots, v_{i-w})|\Phi(v_i)) \tag{20}$$

- 放松了链条的有序要求，把最大化从 v_i 附近看到链条的概率，变成最大化从 v_i 附近看到链条中任意节点的概率，并且进行变形得到式 (21). 注意外侧括号里面从表示有序数组的小括号变成了表示无序集合的大括号；

$$\min \ -\log \Pr(\{v_{i+w}, \cdots, v_{i+1}, v_{i-1}, \cdots, v_{i-w}\}|\Phi(v_i)) \tag{21}$$

- 式 (21) 已经可以将拆分成为一系列两点之间概率的乘积了，那么如何表示 $\Pr(u_i|v_j)$ 呢？可以使用 Softmax

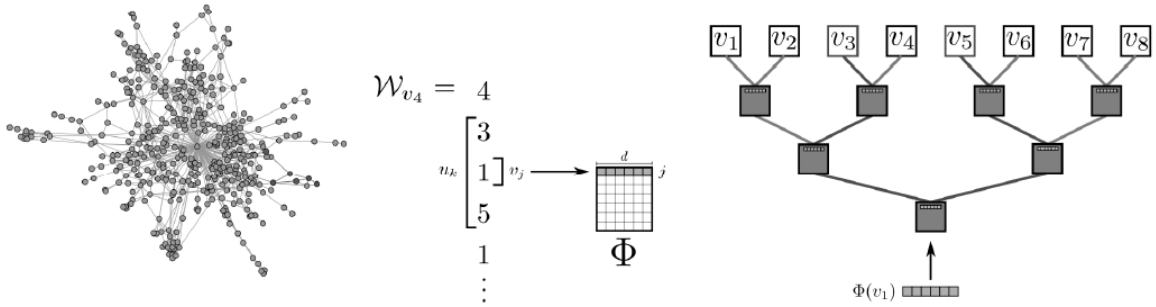
$$\Pr(u_i|v_j) = \frac{\exp(\Phi(u_i) \cdot \Phi(v_j))}{\sum_{v \in \Gamma(v_j)} \exp(\Phi(v) \cdot \Phi(v_j))} \tag{22}$$

最后，DeepWalk 算法可总结如下：

1. 随机选择一种表示矩阵 Φ；
2. 在图中某一个节点 v_i 处进行随机游走，得到一条链 $\mathcal{W}_{v_i}$；
3. 在链 $\mathcal{W}_{v_i}$ 上使用 SkipGram 方法：
   (a) 对 $\mathcal{W}_{v_i}$ 中的某一个点 v_j：
   (b) 在链条上找到 v_j 附近的一系列点 u_k，计算 $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
   (c) 更新表示矩阵：$\Phi \leftarrow \Phi - \alpha \dfrac{\partial J}{\partial \Phi}$
   (d) 对 $\mathcal{W}_{v_i}$ 中的所有点重复 (a)∼(c) 操作
4. 对 G 中所有节点重复 2∼3 操作

Node2Vec

Node2Vec 算法的基本思想与 DeepWalk 是相同的，都是通过随机游走，不断优化。然而完全的随机游走在某些情况下所提供的信息并不充足。Node2Vec 算法的核心是改变随机游走的策略，通过两个参数调制随机游走过程中的倾向。由于随机游走与搜索图的过程有些类似，所以这里我们回顾两个图上经典的搜索策略，借此说明 Node2Vec 的意义。



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

Figure 3: DeepWalk's Algorithm.

- **广度优先搜索 (Breadth-first Search)** 是指从一个点出发，先遍历所有与这个点距离为 1 的点，然后再遍历所有与这个点距离为 2 的点，以此类推。在程序实现上，可以用队列存在遍历的目标
- **深度优先搜索 (Depth-first Search)** 从一个点出发，先尽可能地逐渐增加下一个点与最初点的距离，直到寻找不到满足条件的点后，再向前回溯。

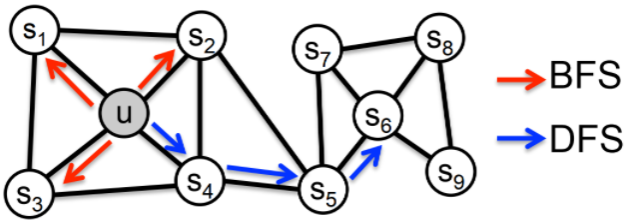


Figure 4: 广度优先搜索与深度优先搜索.

图4展示的就是广度优先搜索（BFS）和深度优先搜索（DFS）的两个例子。在通过随机游走过程中构建节点链条、进而优化表示向量的过程，由于图本身的性质不同，我们应该合理地调整随机游走的策略才能发掘不同图的信息。而且，从图嵌入的角度分析，如果随机游走过程中更倾向于在初始点附近游走，那么得到的表示向量将更多地存有局部关联性信息；而如果随机游走的过程倾向于从初始点出发向远处探索，那么得到的表示向量将更多地存有全局的关联信息。Node2Vec 提出的算法试图解决这一问题。它采用了二阶随机游走策略，即每一次随机游走不仅取决于当前点的结构，还要参考上一步随机游走过程。

假设通过随机行走，我们从 t 来到了 v，那么下一步对于所有与 v 相连的节点 $x \in \Gamma(v)$，采用式(23) 所表示的未归一化的概率进行选取，引入了两个超参数 p, q，来控制随机行走过程中回到上一点与向着更远方向行走的概率。

$$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}, \qquad \alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{23}$$

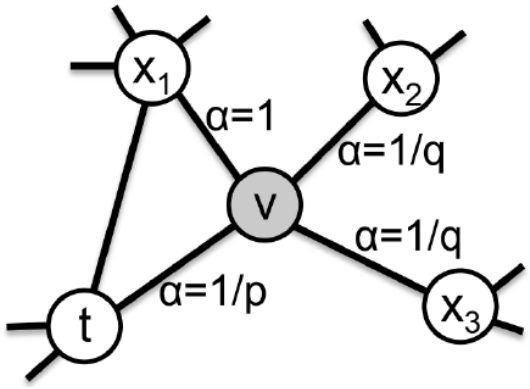


Figure 5: 随机行走策略

(23) 式中的三种情况，分别对应着回到点 t；t，x 直接相连；t, x 不直接相连。注意到由于点 v 的存在，所以 t 和 x 之间的距离最大便是 2. 可以参考图5。

Verse

Verse 的全名是“基于相似性测量的通用图嵌入算法” [20]（Versatile Graph Embeddings from Similarity Measures），它旨在构造一个可以自主选择相似函数的图嵌入框架。它的基本思路与 LINE 相似：最小化图与嵌入得到的表示空间的 KL-divergence.

让我们回顾 LINE，该算法使用了比较简单的 $\hat{p}(i,j) = \frac{w_{ij}}{W}$，然而对 p 的选择上则区分了一阶相似函数与二阶相似函数。不同于此，Verse 允许使用多种 $\hat{p}(i,j)$，却仅仅使用单一的 $p(i,j) = u_i^T \cdot u_j$。在论文中他们展示了三种经验函数 $\hat{p}$：Personalized PageRank、Adjacency similarity、SimRank。

- **Personalized PageRank**

  源自 PageRank，可理解为一种给网页评分的方法。可通过 (24) 式的递归方程计算。其含义是模拟一个网站上随机浏览的人，他每次以一定概率跳转到下一个网页或者回到初始状态。和 DeepWalk、node2vec 的相同之处在于都模拟了随机行走，并且 [20] 证明了这里的衰减系数 α 与 DeepWalk(node2vec) 中的链条长度有着很大关系。不同之处在于，通过 (24) 这个递推式，我们可以直接多次迭代计算出它的稳定分布 π_s，其向量中每一个值的意义就是从初始分布经过多次迭代后，来到对应的点的概率。

$$\pi_s = \alpha s + (1-\alpha)\pi_s A \tag{24}$$

- **Adjacency similarity**

  与 LINE 中的 $\hat{p}$ 是类似的，当两个点之间存在连线时，使用归一化后的值作为其概率；若两个点不连通，则值为 0。

- **SimRank**

  是用作测量两点之间的联系程度的方法，它逐个考察两个点 u, v 的邻居对 x_u，x_v，如果邻居之间的关联很大，那么有理由认为这两个点之间的关联也大。由此，可以用 (25) 迭代得到结果。式中 $I(u), I(v)$ 表示那些存在指向节点 u, v 的点。在一张不定向图里，他们应当等价于 $\Gamma(u), \Gamma(v)$。

$$SimRank(u,v) = \frac{C}{|I(u)||I(v)|} \sum_{i=1}^{|I(v)|} \sum_{j=1}^{|I(u)|} SimRank(I_j(u), I_i(v)) \tag{25}$$

正如其名字所展示的那样，Verse 允许使用多种经验函数来表示图中节点之间的信息，然而其代价就是计算过程的复杂程度提高了。或许作为一种补偿，Verse 定义的相似函数是表示向量之间简单的内积，如式 (26)，其中 W 是将节点从 one-hot 编码映射到嵌入的向量空间中某一向量的算符，由于 one-hot 编码用的是正交的单位向量，所以可知每一个映射结果都是算符 W 中的一个行向量或列向量。

$$p(u,v) = W_u \cdot W_v \tag{26}$$

但是根据 [19], 这样定义的相似函数只能提供一阶相似。Verse 采用的方法是，定义另外一个 W'，给 $p(\cdot,\cdot)$ 中的两个节点以不同的表示策略，如式 (27)。这样可以打破相似函数的对称性，然后同时优化这两个算符，从而达到二阶相似的效果。

$$p(u,v) = W_u \cdot W'_v \sim \hat{p}(u,v) \tag{27}$$

这种对于一条边的两个节点训练两个不同的表示向量的方法，可以给出不对称的内积，从而更满足一类 $p(u,v)$ 与 $p(v,u)$ 不相等的启发式评分。受此启发，STRAP [25] 进一步研究了要如何训练表示矩阵。

STRAP

对于 DeepWalk、Verse 一类嵌入算法所用到的相似函数，若对从一个节点出来的所有相邻节点进行求和，相似函数的和是归一的。然而用这样相似函数将无法保存某些图所具有的信息，或者无法区分指向一个点以及从这个点指出的边的差异，如图：

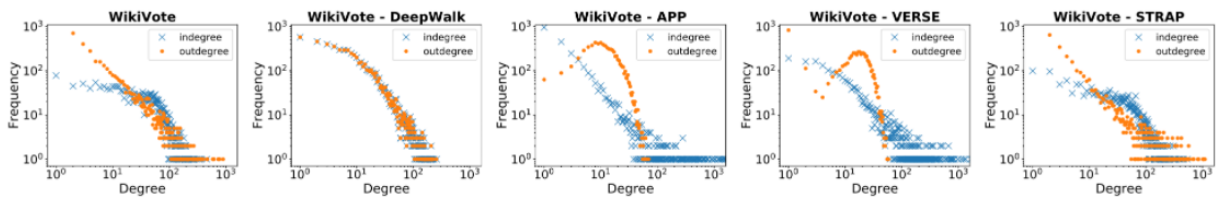


Figure 6: WikiVot 在不同嵌入下，进出度的分布

此外，从训练的角度：DeepWalk、Node2Vec 等算法用 $s_u \cdot s_v \sim p(u,v)$ 来训练，然而 s_u, s_v 的内积是对称的，可往往 $p(u,v)$ 和 $p(v,u)$ 并不相等，所以在优化过程中往往会引入冲突的优化目标，导质优化效果不好；而有些算法，比如 Verse 使用不对称的表示向量 $W_u \cdot W_v^{'}$，这避免了冲突的优化目标，这又引入新的问题——判断两个节点是否存在链接时，到底使用 $W_u \cdot W_v'$ 还是 $W_v \cdot W_u'$ ？

因此，STRAP [25] 引入了“转置图”的概念，如果在原始图中存在一条从 u 指向 v 的边，那么在这个转置图里就存在一条从 v 指向 u 的边。我们用 G^T 来表示转置之后的图，然后延续 Verse 的想法，不过用 (28) 式代替 (27) 式.

$$W_u \cdot W_v' \sim \hat{p}^T(v,u) + \hat{p}(u,v) \tag{28}$$

2.3.3 图嵌入算法中用到的优化方法

Jiezhong Qiu 等在论文 [16] 里说明了，上述一系列图嵌入算法都是在显示或隐式地优化某个矩阵，并用其表示图中的各个节点。由于真实世界中图存储的复杂社交网络数据往往体量巨大，所以图嵌入的算法常常要考虑如何减少复杂的优化过程，这里列出了论文中用到的几种优化方法。

Noise constrain estimation

NCE 的核心想法大致可以理解为：一个好的模型应该可以和噪声的分布有着显著的差异，可以通过 logistic regression 来进行区分。想象我们有一个数据分布 $\mathcal{Q}$ ，我们随机地用来自另一个分布 $\mathcal{P}$ 的数据点混杂其中，然后用一个 logistic regression 模型去最小化这个 log-likelyhood:

$$\mathcal{L}_{NCE} = \sum_{\substack{u \sim \mathcal{P} \\ v \sim sim_G(u,\cdot)}} \left[\log \Pr_W(D=1|sim_E(u,v)) + s\mathbb{E}_{\tilde{v}\sim\mathcal{Q}(u)} \log \Pr_W\left(D=0|sim_E\left(u,\tilde{v}\right)\right)\right] \tag{29}$$

NCE 的一大优势在于，它可以直接寻找未归一化的概率分布的参数，这弥补了极大似然估计的不足。

Negative Sampling

相比于 NCE，负采样（Negative Sampling, NEG）是基于采样来进行二元逻辑回归，不过进行了一定的简化。它最初在 Word2Vec 的论文中被提出 [13]。传统的 NCE 可以视为近似地最大化 softmax 的对数概率，并且使用了额外的噪声分布。NEG 假设训练的样本具有高质量的向量表示，在遇到某个点 v 时，可以用其它没有遇到的点代替 NCE 中的噪声，而寻找这些点的过程就被称为负采样。

$$\log \sigma\left({v'_{w_O}}^T v_{w_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma\left(-{v'_{w_i}}^\top v_{w_I}\right)\right] \tag{30}$$

2.3.4 用图嵌入得到表示向量进行链接预测

图嵌入帮助我们获得了节点的向量表示，由此我们可以使用 Logistic regression 分类器判断边是否存在。虽然理论上我们可以将两个节点的表示“拉直”成一维向量作为分类器的输入，然而实际操作中

我们有更好的选择，比如可以先用某种算符对两个向量进行操作，用得到的结果作为链接预测的输入。常见的算符如下：

- Average：即取平均

$$(\boldsymbol{a}+\boldsymbol{b})/2$$

- Concat：即我们之前所谓的“拉直”

$$[a_1,\cdots,a_d,b_1,\cdots,b_d]$$

- Hadamard：

$$[a_1\cdot b_1,\cdots,a_d\cdot b_d]$$

- Weighted L1:

$$[|a_1-b_1|,\cdots,|a_d-b_d|]$$

- Weighted L2:

$$\left[(a_1-b_1)^2,\cdots,(a_d-b_d)^2\right]$$

把两个节点的表示向量进行上述操作后，就可以得到两个节点所确定的边的表示，可以以此作为分类器的输入训练一个判断链接存在概率的模型。然后类似启发式方法，我们选定阈值，预测那些阈值以上的边存在链接。在图嵌入的学习中，我们是在利用边的启发式关系训练节点的低维表示，然后进一步用节点的表示计算边的表示，从而进行链接预测。或许可以跳过这一步，直接训练边的低维表示作为输入，这值得进一步的研究。

2.4 基于图神经网络的链接预测

现在已有的诸多链接预测方法中，无论是传统的启发式方法；还是使用嵌入寻找点的低维表示，再用得到的向量进行链接预测的输入，都需要人为的设定标准。进行启发式的推断，即考虑：什么样的点之间更有可能存在链接。

这类方法的弊病在于，存储不同信息的图网络中，存在链接的两个点的类型可能有很大不同。因此不同的网络必须要使用相同的启发式算法才能得到更加准确的预测结果。在 [11] 中，作者比较了不同数据集上超过 20 种启发式算法的揭发，发现它们都不能持续保持相同的预测效果。因此可见，启发式算法需要人为地根据图网络的拓扑结构选取合适的算法。

基于这些问题，设计一个泛化性更好的算法，让其主动地从一个图网络中学习拓扑结构并进行链接预测，便成为了一种自然的想法。受近两年非常火热的图神经网络（GNN）发展带来的启发，Muhan Zhang 等人提出了 SEAL [27] 算法，将节点附近的封闭子图作为图神经网络的输入，从而进行链接预测。

在本节，我们首先简单地对图神经网络进行介绍，然后着重介绍 SEAL 算法的核心内容与原理；由于图神经网络进行链接预测是使用局部的一个子图作为输入，我们将在第三部分证明这样做的合理性。

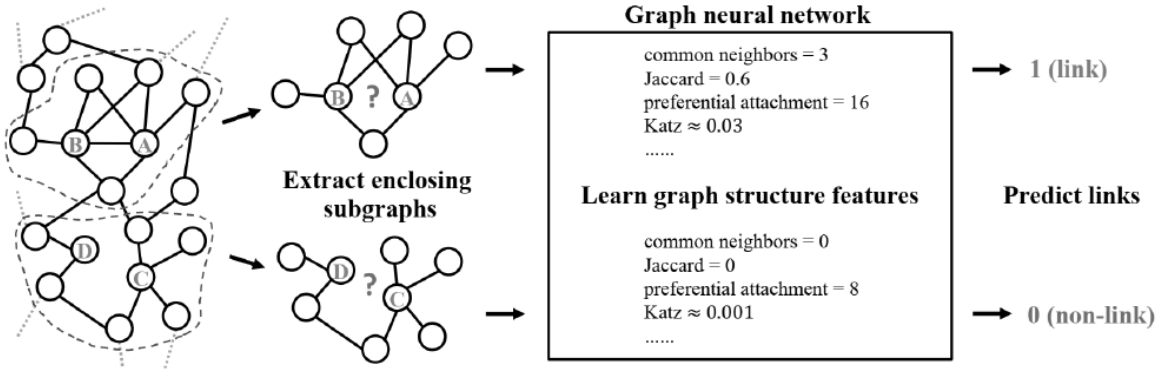


Figure 7: SEAL: 提取封闭子图并使用 GNN 进行学习

2.4.1 图神经网络

近年来，深度学习领域关于图神经网络（Graph Neural Networks，GNN）的研究热情日益高涨，图神经网络已经成为各大深度学习顶会的研究热点。GNN 处理非结构化数据时的出色能力使其在网络数据分析、推荐系统、物理建模、自然语言处理和图上的组合优化问题方面都取得了新的突破。

在 SEAL [27] 中，研究者使用的 Deep Graph Convolutional Neural Network（DGCNN），该图神经网络使用 $(\boldsymbol{A}, \boldsymbol{X})$ 作为输入，其中 A 是邻接矩阵，X 存储的是节点信息，其每一行表示与一个点相关的特征向量。DGCNN 的卷积层按如下方式进行卷积：

$$\boldsymbol{Z} = f\left(\tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}\ \boldsymbol{X}\ \boldsymbol{W}\right) \tag{31}$$

其中 $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ 是由邻接矩阵加上单位阵，$\tilde{\boldsymbol{D}}$ 是对角阵，每个对角元的值等于所对应点的入度。$\boldsymbol{W}$ 可以看作一个线性变换，将矩阵的信息 $\boldsymbol{X}$ 经过变换后，通过前面的 $\tilde{\boldsymbol{D}}^{-1}\tilde{\boldsymbol{A}}$ 的作用，让信息在每个节点周围进行传播。这类似于传统欧氏空间的卷积操作。通过卷积层，DGCNN 可以提取出图的一些特征。之后采用排序池化的策略，在将特征 (每个特征描述一个顶点) 放入传统的一维卷积和密集层之前，以一致的顺序对它们进行排序，这里采用的是 WL 连续上色法对节点进行排序，图8展示了 DGCNN 的整体结构。

这里只是作为 SEAL 方法的背景知识，对图神经网络进行的一个简单介绍。图神经网络近年来的发展极为迅速，关于其更多知识可以参考 [17] [24] [9].

2.4.2 SEAL

SEAL 的做法是：先在某个链接附近提取出一个封闭的子图，然后用该封闭子图的邻接矩阵作为输入，通过训练一个图神经网络进行链接预测。由于提取的封闭子图可以提供每个链接周围的拓扑结构，神经网络便可以自主的找到合适的评判标准来进行预测，从而让人们不必人为地寻找启发式方法。

编号

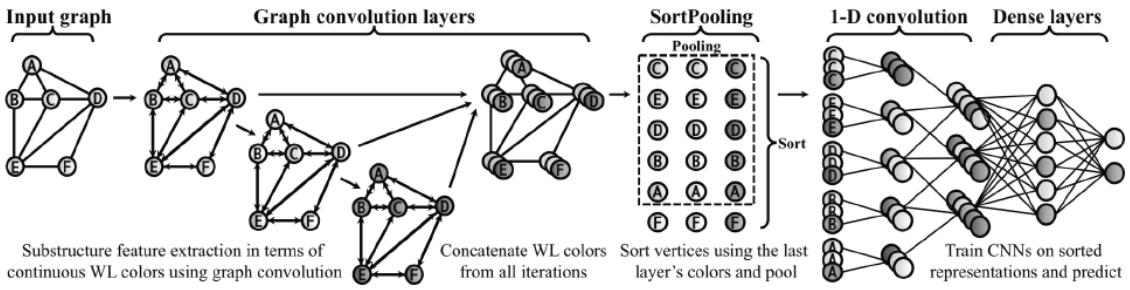


Figure 8: DGCNN 的整体结构

提取封闭子图，并使用 GNN 进行链接预测的思路很清晰，然而其核心在于如何构建子图的邻接矩阵。由于我们要提取不同节点对附近的封闭子图，并且以此作为图神经网络的输入，因此必须在提取所有子图的时候都采用某种不变的策略对子图中的点进行编号，这样作为输入的邻接矩阵才可能产生一个有意义的模型。在连接预测过程中使用的编号策略应该满足：

- 在子图中具有相似结构的点应该被标上相似的序号
- 该标号要能在子图中区分原始的“目标链”，并在排序时保持拓扑的方向性。

在这里我们先介绍一种和 SEAL 想法类似，只不过采用的是全连接神经网络的链接预测算法：WLNM [26]。WLNM 也需要在封闭子图上进行编号，它采用的策略是，让连通性高的节点具有值比较大的标号，用相同颜色表示值相同的节点，并根据其邻居的标号大小将相同标号的节点“展开”成为不同的编号。

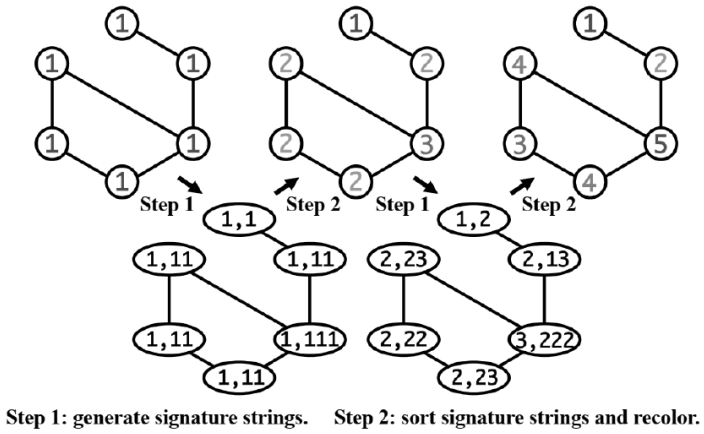


Figure 9: WLNM 所用的编号策略

应该强调的是，封闭子图是天然地具有内在方向性的：中心位置是“目标链”，越偏离中心对应的点与“目标链”的两端之间的距离越大。因此，SEAL 基于封闭子图的想法，采用了另一种编号策略：

对于两个节点 x, y, 其封闭子图中与这两个节点距离之和更小的点拥有更小的编号，编号随距离增加而增加。由此可以得到一种用来编号的哈希函数：

$$f(i) = 1 + \min(d_y, d_y) + (d//2)[(d//2) + (d\%2) - 1] \tag{32}$$

式中 $d_x := d(i, x)$, $d_y := d(i, y)$, $d := d_x + d_y$, $(d//2)$ 和 $(d\%2)$ 代表取商与取余。使用式 (32) 获得编号之后，就能进一步获得邻接矩阵。也可以直接用编号得到子图的 one-hot 编码，作为 X 的输入。不过为了获得更多的潜在信息，我们可以进一步构造 X。

构造信息

为了获得图的潜在信息作为 GNN 的输入，SEAL 用图嵌入的方式得到了不同节点的表示，并把其放于 X 中。然而由于图嵌入得到的表示向量天然地会带有链接存在与否的信息，实验中发现，GNN 可以快速地找出这些信息，并以此作为链接预测的判据。这很不利于泛化。

SEAL 采取的策略是负注入（negative injection），在进行图嵌入的过程中掺杂一些原本不存在的边，让 GNN 不能完全从表示向量中获得链接存在与否的信息，由此来强迫其学习图的结构特征，获得更好的链接预测结果。

2.4.3 γ-decay theory

论文 [27] 中为了证明从局部获得的封闭子图可以包含链接预测所需要的信息，对此进行了一些理论分析。他们提出了一种 γ-decaying 启发式方法，并证明了一些主流的启发式方法都可以归进这个体系。他们定义的 γ-衰减启发式方法具有如下形式

$$\mathcal{H}(x, y) = \eta \sum_{l=1}^{\infty} \gamma^l f(x, y, l) \tag{33}$$

其中 γ 是一个 0 到 1 的衰减系数，η 是一个正常数或者 γ 的有上界的函数，f 是一个非负函数. 论文中证明，如果有式 (33) 并且 $f(x, y, l)$ 在 $G^h_{x,y}$ 中是可以被计算的，对于 $l = 1, 2, \cdots, g(h)$, 其中 $g(h) = ah + b$ with $a, b \in \mathbb{N}$ and $a > 0$，那么 $\mathcal{H}(x, y)$ 可以从 $G^h_{x,y}$ 进行近似得到，并且其近似误差随着 h 以指数形式衰减。

$$f(x, y, l) \leqslant \lambda^l \text{ where } \lambda < \frac{1}{\gamma} \tag{34}$$

2.5 小结

本章作为链接预测的一个简单综述，将已有的链接预测方法分为三类：启发式预测、基于图嵌入的预测、基于图神经网络的预测，并分别介绍了相关的算法。启发式预测方法主要注重于人为的设计评分函数，通过数据的各种特性来提出我们的指标，因此，启发式方法的针对性较强，但通用性较低。基于图嵌入的预测，主要通过得到节点的特征嵌入来解决链接预测问题，但这种方法具有较强的先验知识分布，不利于整体的泛化。基于图神经网络的方法目前比较主流，主要归于与图神经网络强大的解释表示能力，现在主流的解决链接预测的方法主要有两类，分别为 GAE [7] 和 SEAL [27]，我们将在下一节的实验报告中分别提及。

3 实验报告

3.1 电信网络数据介绍

电信网络是支持个人通信以及商业组织之间的通信的关键。为了确保大规模电信网络的正常运转，我们需要对电信网络中出现的告警进行及时处理，从而实现电信网络中的故障管理，其目的也是保证电信网络的安全性与可靠性。但是由于电信网络庞大，其产生的告警数量也非常多，这给告警的处理带来了很棘手的问题，为了确保故障管理的高效性，我们需要识别出众多告警数据中具有诱发性的告警，优先处理。然而，由于数据的缺失与电信网络在搜集数据中具有很多的不确定性，根据数据所构建的拓扑图中具有很多缺失的边，此时我们需要根据已有的拓扑图及告警数据进行电信网络图上的链接预测，来挖掘网络中具有潜在可能性能够相连的边，从而实现电信网络的拓扑还原。

我们使用的数据来自于印尼的大型电信设备网络数据，采集的数据的起始时间分别维 2019 年 4 月 12 日到 4 月 16 日，总跨度维为 5 天。印尼数据主要包括两个部分：告警数据 (微波域和无线域) 和拓扑数据。数据集中有超过 60 万条告警，分别属于 300 多个种类，由不同的设备发出。

Alarm Name	Domain	Alarm Source	Occurrence Time	Clear Time
Alarm 1	ran-4g	Source 1	2019-04-12 10:40:23	2019-04-12 10:40:29
Alarm 2	microwave	Source 2	2019-04-12 10:40:24	2019-04-12 11:30:44
Alarm 3	ran-2g	Source 3	2019-04-12 10:40:26	2019-04-12 10:40:36
...	...	...	...	...

Figure 10: 告警数据字段和统计信息

- 告警数据
  用到的告警数据来自于微波域和无线域，图10是告警数据集中所包含的字段和统计信息。

- 拓扑数据
  拓扑数据由工程师收集的一条条信息流路径组成。用信息流路径可以还原出站点拓扑。主要字段信息与统计信息如图11所示。

Path Id	Device Name	Device Type	Path Hop
1	Device 1	Router	0
1	Device 2	Microwave	1
1	Device 3	RAN	2
2	Device 1	Router	0
2	Device 4	Microwave	1
2	Device 5	RAN	2
...	...	...	...

Figure 11: 拓扑数据字段和统计信息

告警数据中包括告警名称、告警所属的范围、告警产生的源头、告警产生的时间以及告警消失的时间。拓扑数据中包括网络中的一些节点、它们所属的类别以及在网络中的级别，我们可以根据这个拓扑信息来构建一张拓扑图，从而为后续的分析提供原始数据。由于电信网络的特殊性，其具有典型的树型结构，我们可视化部分网络节点如图12：

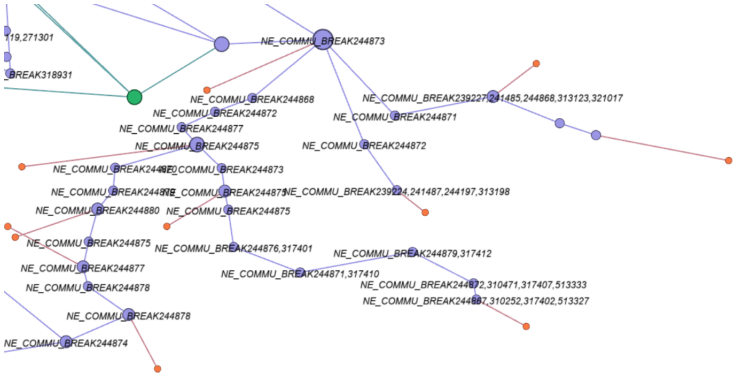


Figure 12: Telecom Network

在获取警报日志后，需要对数据进行预处理以滤除一些虚假警报。这是根据电信网络专家的建议完成的，并允许对警报相关性进行更精确的分析并减少计算所需的时间。首先，将在短时间内（根据领域专家的推荐为 5 分钟）内重复出现在设备中的所有警报合并。其次，一些持续时间很短的重复发生的警报被过滤掉，因为它们被认为是无意义的。第三，丢弃信息不完整的警报（例如空的警报源字段）。

在以往的研究中，我们通过分析历史告警数据来寻找多个告警之间的模式关系，但是大多数的研究仅仅将告警数据视为时序告警数据，这忽略了告警设备之间的拓扑关系，从而会丧失某些潜在的信息，得到不准确的结果。

于是我们将电信告警数据建模成动态网络图，告警数据可以根据信息流进行传播。通过有向图的形式，将图的顶点视为设备，边视为信息流传播的方向。根据拓扑数据，我们将电信网络中的设备构建成一个有向图。然后我们需要将告警信息放在网络中，我们将网络中的设备名与告警发生的设备名一一对应，从而将告警映射到网络中，最终我们得到的网络结构如图13所示，这表示网络中的一个特定节点的特征。这种基于图形的结构指示告警如何随时间变化，是一种动态属性图。需要注意的是，在映射过程中，没有映射到拓扑中任何设备的告警都会被丢弃。

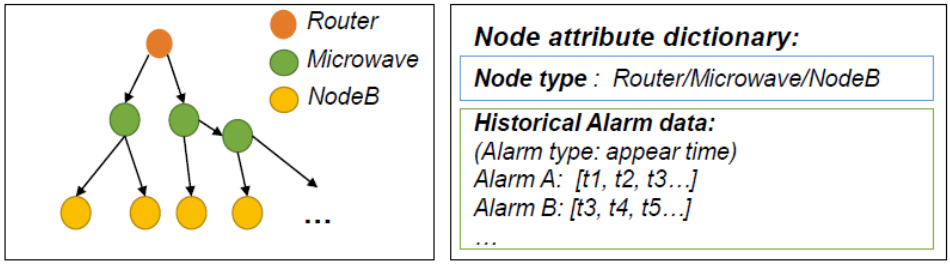
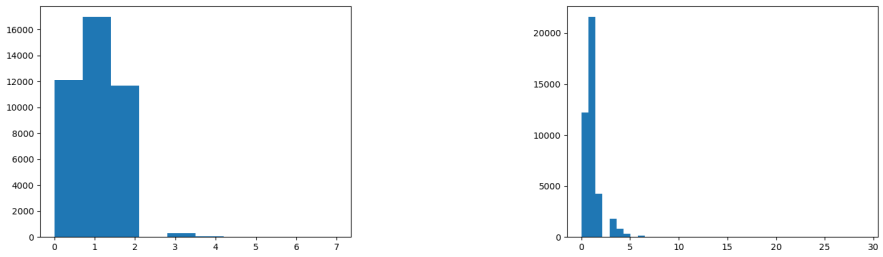


Figure 13: The recovered network topology (left) with alarm attributes (right)

我们进一步对于构建好的拓扑图进行分析。拓扑图中共有 41143 个节点以及 41142 条有向边，在对告警数据进行预处理之后，我们得到网络中的告警类型共有 240 种，低于原先的 300 多种。考虑到这是一个电信网络图，幂律现象应该是一个比较明显的特征，我们首先统计图中节点的入度和出度，如图14。



(a) 入度　　(b) 出度

Figure 14: 节点的度信息

我们从节点的度可以看出，大部分的节点入度和出度都非常小，这说明这是一个非常稀疏的图。只有极少的节点具有比较大的出度，这说明该节点可能为某个中心设备。

同样的，网络图中每一个节点对应的设备都有相应的类别标签，41143 个节点共分为 3 类，分别为 Router、Microwave、NodeB，三种不同的设备分别有 507、24515、16121 个。信息流传递的方向为从 Router 到 Microwave 到 NodeB，所以这是一个树型的网络结构，但由于 Microwave 之间可以相互连接，所以又不是一个非常严格的树。因此，这种树型结构的图就构成了电信网络图的基本结构。

同样的，我们所关心的还有网络中连接节点之间的边，之前所说这个网络为树型结构，所以其边的存在大多为两个不同类型的节点之间，我们统计各种类型的边在网络中出现的次数如表1所示。

Table 1: Number of Edges among Different Devices

Source device	Target device	Number of Edges
ROUTER	ROUTER	0
ROUTER	MICROWAVE	1428
ROUTER	NODEB	19
MICROWAVE	ROUTER	573
MICROWAVE	MICROWAVE	23541
MICROWAVE	NODEB	4116
NODEB	ROUTER	3
NODEB	MICROWAVE	12003
NODEB	NODEB	0
total		41683

3.2 实验结果

实验中我们直接使用已经构建好的网络图。目前研究领域内常用的链接预测的方法主要是图自编码器和 SEAL，图自编码器通过重建邻接矩阵的方式来训练得到较为理想的节点特征，从而实现链接预测的功能；SEAL 通过抽取边的邻居子图的方式来训练，并通过特定的 DRNL 的贴标签的方式来为子图中的节点赋予新的特征，通过图神经网络的训练来达到链接预测的效果。

对于已有的网络图，我们从其中能够得到节点之间的连接关系，但是每一个节点的特征信息是以字典形式存储在节点中的，并不能被直接利用。因此，我们需要根据已有的信息构建出可用的节点特征向量。一个比较常见的想法是将 240 种告警类型进行独热编码 (one-hot encoding)，若该设备上出现过此类告警，则令为 1，反之为 0。这样我们对每一个节点会得到一个 240 维的向量作为其特征。同时，若利用到告警出现的次数，我们将时间轴划分成无数个时间戳，在每一个时间戳内用独热编码对节点进行特征表示，然后将所有时间戳的特征相加，得到新的特征，相当于是将这个节点上某种告警出现的次数作为它的特征，我们将这种编码特征的方式称为 count。为了验证 one-hot 编码和 count 编码的有效性，我们设置对比试验，用随机数的方式产生起始特征，同样得到一个 41143×240 的特征表示。这三种方式的特征编码将会用于之后的实验中。

由于我们的任务是链接预测问题，其本质上是一个二分类问题，所以在评估模型性能的时候，我们选择了二分类问题常用的评价指标，分别为 AUC(Area under curve) 和 AP(Average Precision)。

3.2.1 图自编码器 (Graph Auto-Encoder)

简介

对于图自编码器 (GAE)，分为编码器和解码器两个部分。编码器通过某种方式编码图中节点的特征 (隐式表示)，这些特征然后通过解码器计算出图中某条边存在的概率，从而达到重建邻接矩阵的效果。

图自编码器的选择图神经网络 (GNN) 作为编码器，来得到节点的特征表示，这个过程可以用一行简短的公式表达：

$$Z = GNN(X, A) \tag{35}$$

将 GNN 视为一个函数，然后将 X 和 A 作为输入，输入到 GNN 这个函数中，输出 $Z \in \mathcal{R}^{N \times f}$，$Z$ 代表的就是所有节点的隐式表示，或者说特征。

图自编码器使用内积 (inner product) 作为解码器来重构原始的图：

$$\hat{A} = \sigma(ZZ^T) \tag{36}$$

上式中，$\hat{A}$ 就是重构出来的邻接矩阵。

一个好的隐式表示 Z 就应该使重构出的邻接矩阵与原始的邻接矩阵尽可能相似，因为邻接矩阵决定了图的结构。因此 GAE 在训练过程中，采用交叉熵作为损失函数：

$$\mathcal{L} = -\frac{1}{N} \sum y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \tag{37}$$

上式中，y 代表邻接矩阵 A 中某个元素的值，$\hat{y}$ 代表重构的邻接矩阵 $\hat{A}$ 中相应元素的值，我们希望重构的邻接矩阵与原始的邻接矩阵越接近、越相似越好。

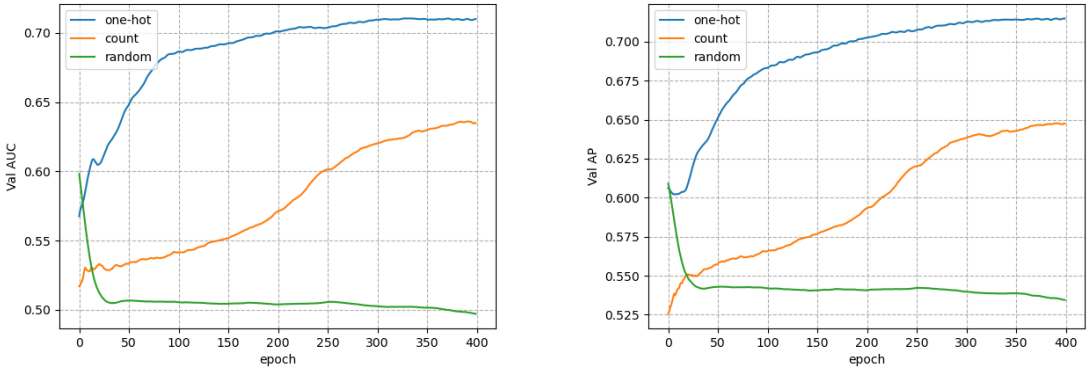
实验结果

在做链接预测的实验中，我们需要划分训练集、验证集以及测试集，我们随机的从网络图中划掉 15% 的边作为验证集和测试集 (验证集：5%，测试集：10%)，其余的 85% 的边作为训练集，这些边均为正样本。同样，我们使用负采样的方法从图中选取不存在的边作为负样例，并保持正边数等于负边数。

在训练集/验证集/测试集划分完之后，我们需要根据训练集的边来构建训练时用到的图，而不使用原来的图，这是因为原始的图中含有验证集和测试集中的正边，因此要将其从原始的图中去除。为了保证实验结果的可比较性，我们使用相同的实验环境，训练轮次 400，学习率 0.0001，优化器使用 Adam，并在同样的 GeForce RTX 3090 上使用 GPU 训练，并使用 AUC(Area Under Curve) 和 AP(Average Precision) 进行模型性能的度量。

针对图自编码器中编码器的可选择性，我们分别选取 GCN [8]，SGC [23]，GAT [21] 这三种图神经网络的模型作为编码器，使用内积作为解码器，分别进行实验。

当使用 GCN 作为 GAE 的编码器时，我们使用两层的 GCN 模型，同时将输出特征的维度设置为 32，分别使用不同的起始特征编码的方式来进行实验，得到验证集上的 AUC 和 AP 如图15所示。



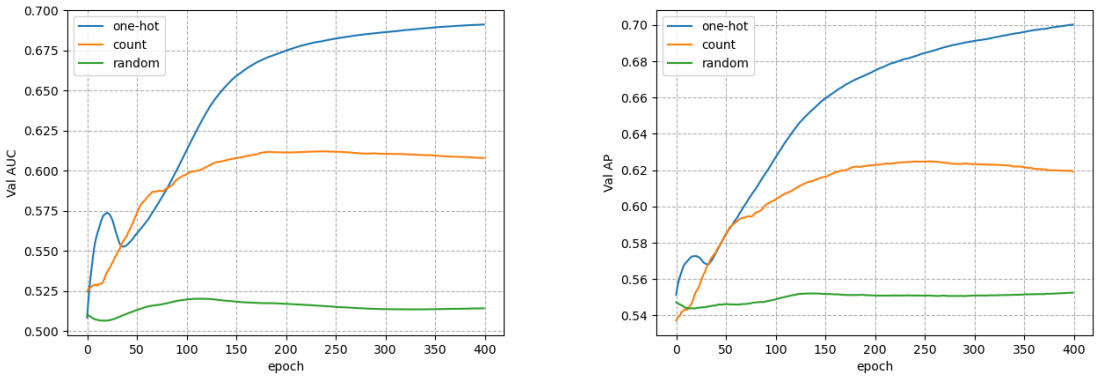
(a) AUC on validation set

(b) AP on validation set

Figure 15: Metrics on the validation set(GCN as Encoder)

从图15中可以看出，使用 GCN 作为编码器，inner product 作为解码器时，one-hot 初始编码的效果时最好的，其在验证集上的 AUC 和 AP 达到了 0.7100 和 0.7148。其次是 count，其 AUC 也超过了 0.6，最差的是 random 随机编码，大概在 0.5 左右，效果等价于随机猜。这很可能是因为在划掉了验证集和测试集以后，网络的结构有一定程度的破坏，导致信息的传递出现阻碍，从而使得网络的拓扑结构无法为消息传递提供高效的支持，于是在 random 的特征编码下，特征的无效更新使得模型的表现很差。

之后使用 SGC 作为图自编码器的编码器，SGC 与 GCN 的不同之处在于，其在图神经网络消息聚合的那一步中并没有使用激活函数，这使得模型的非线性程度下降。在这里，我们同样使用两层的 SGC, 同样使输出特征维度为 32，得到实验结果如图16所示。



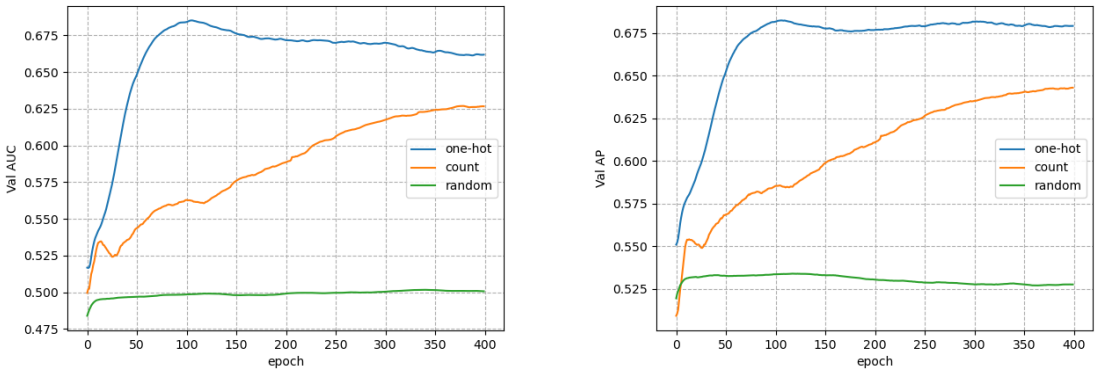
(a) AUC on validation set

(b) AP on validation set

Figure 16: Metrics on the validation set(SGC as Encoder)

从图16中可以看出，使用 SGC 得到的结果与使用 GCN 得到的结果并没有太大的差距，其在验证集上的 AUC 和 AP 分别达到了 0.6912 和 0.7002。综合来说，使用 GCN 作为编码器效果更胜一筹，这可能是因为 SGC 相比于 GCN 减少了非线性模型的效果，从而使得精度得以下降。

当使用 GAT 作为编码器时，我们使用 multi-head attention 机制作为我们的模型，我们将网络层数设置为 2，第一层网络的 head 设置为 8，隐层特征数为 8，concat 设置为 True，这样对于第一层网络，我们得到的是一个 64 维的隐式向量，第二层网络，我们设置 head=1，最终得到一个 32 维的特征表示。与前面的实验环境相同，我们得到结果如图17所示。



(a) AUC on validation set

(b) AP on validation set

Figure 17: Metrics on the validation set(GAT as Encoder)

从图17中可以看出，与前两种编码器相同的是，都是使用 one-hot 作为初始编码效果最好，count 次之，random 最差。图中表明使用 one-hot 初始编码时，很快便能达到较高水平的 AUC，随之便出现

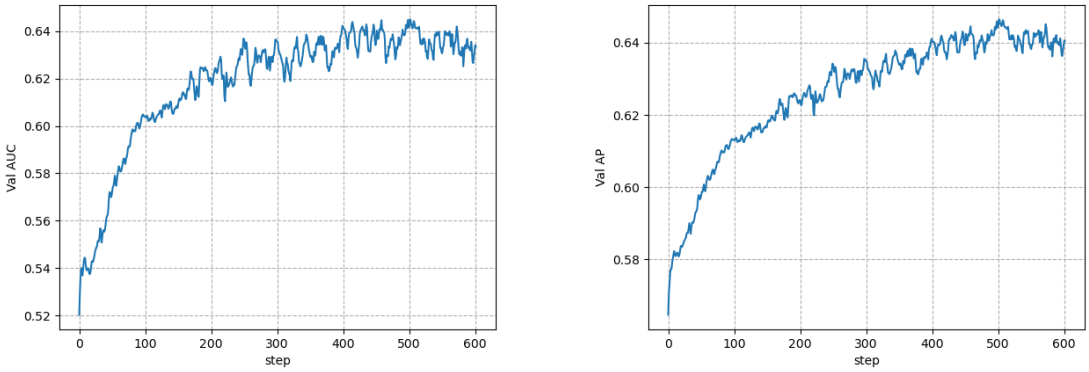
了过拟合现象，然而，使用 count 作为初始编码时，无论是 AUC 还是 AP 的增长速度都较慢，且从图17中可以看出，其模型尚未训练完成，还处于欠拟合的状态，仍然需要更多的训练轮次，因此，后续的工作可以增加训练的轮次，从而重新衡量模型的效果。

综合 GCN、SGC 以及 GAT 三种图神经网络作为编码器，以及 one-hot、count、random 三种不同的特征编码方式，我们对所有结果进行比较如表2所示。

Table 2: Metrics of different configurations

Encoder	embedding	Val AUC	Val AP	Test AUC	Test AP
GAE(GCN)	one-hot	0.7100	0.7148	0.7126	0.7149
	count	0.6349	0.6475	0.6259	0.6391
	random	0.4970	0.5344	0.5979	0.6042
GAE(SGC)	one-hot	0.6912	0.7002	0.6817	0.6856
	count	0.6088	0.6205	0.6133	0.6269
	random	0.5141	0.5525	0.5082	0.5445
GAE(GAT)	one-hot	0.6622	0.6791	0.6751	0.6707
	count	0.6266	0.6429	0.6373	0.6429
	random	0.5007	0.5276	0.5115	0.5431
HGCN	one-hot	0.6449	0.6465	0.6446	0.6484

表2清晰的展示了不同模型不同初始特征的表现，对于编码器来说，这三者最好的是 GCN，对于特征编码来说，最好的是 one-hot 编码。对于三种编码器来说，它们的模型表现在三种编码方式上趋于一致，均为 one-hot 优于 count，优于 random。而 GCN 的实验证实最优性可以认为是 SGC 模型过于简单，而 GAT 模型过于复杂所导致。但总体而言，使用 GAE 模型做链接预测得到的效果并不太好，与研究领域的 SOTA 的表现还有一定差距。



(a) AUC on Validation Set

(b) AP on Validation Set

Figure 18: Val AUC and AP(HGCN as encoder)

鉴于华为提供的数据集是大型电信网络的数据集，在前文的介绍中提到过这种数据集往往展现出树型的结构，而最近研究领域内新兴的双曲图神经网络对于具有树型结构的网络具有很好的效果，因此我们使用 Hyperbolic Graph Convolutional Neural Network 作为编码器来训练得到节点的特征表示，使用与前面相同的实验配置，得到的结果如图18所示。

我们从图中可以看到，当使用 HGCN 作为编码器时，在验证集上得到的 AUC 仍然不是很理想，大概在 0.6449 左右，这说明仅仅具有树型结构对于在双曲空间进行消息传递并不能起到关键性的作用，而节点的特征的缺失对于模型性能的提高具有较大的影响，后续的工作应该聚焦于如何利用已有的信息去构建出较为理想的特征。

3.2.2 SEAL

SEAL 作为另一种通用的链接预测方法，与 GAE 不同的是，SEAL 通过抽取目标边的封闭子图来进行预测，其获得的不是节点的特征表示，而是对于某一条特定的边的封闭子图的特征表示，并利用这个特征来对链接进行预测。SEAL 的核心思想就是 learning from Subgraphs, Embeddings and Attributes for Link Prediction。SEAL 方法在很多数据集上都实现了 SOTA 的效果，在链接预测领域具有非常广泛的应用。

SEAL 利用了网络图的拓扑结构，让其主动的从一个图中学习拓扑结构并进行预测。本文利用 SEAL 进行链接预测的主要步骤如下：

1. 训练集、验证集、测试集的划分
2. 针对集合中的每一条边从训练图中抽取 h 阶封闭子图
3. 子图中节点特征的构建 (可选择，DRNL，Node2Vec，或者其他合理的特征)
4. 将此视为一个二元图分类问题，利用图神经网络对训练集进行训练。

为了考虑不同训练集/验证集/测试集的比例对于实验结果的影响，我们设置两组不同比例的试验方案，分别设置为训练集：85%、验证集：5%、测试集：10% 以及训练集：70%、验证集：10%、测试集：20%。对于训练集、验证集和测试集中的每一条边，无论是正边还是负边，我们均从训练图中抽取该边的 2 阶封闭子图，论文 [27] 中提出 γ-decay theory 证明从局部子图中可以包含链接预测所需的信息，只需要一个很小的 h 阶封闭子图就可以对链接预测的性能有很高的保证。

在抽取子图的过程中，我们需要构建子图中节点的特征信息，若原始图中含有特征，则可以把这些特征直接赋予子图中的节点，但实验证明，这些原始特征并不能给链接预测的性能带来提升，因为这些特征并不包含子图中的拓扑信息，因此在训练时与其他的链接预测方法并无太大差异。论文 [27] 提出了一种构建子图中拓扑特征的方法：Double Radius Node Labeling(DRNL)，DRNL 考虑重新给子图中的每一个节点编号，具体编号规则见图19，对子图中的每一个节点编号之后，我们通过独热编码的形式对每一个节点进行特征编码，这样得到的特征便具有了子图的拓扑结构信息。

这里有一个细节就是，对于不同的边，其构建的 2 阶封闭子图的规模不同，因此，在对节点进行编号时，其最大编号也不同，为了在后续使用图神经网络统一训练，我们在进行独热编码时需要将所有子图中的特征维度设置为相同值，即为所有子图中的最大节点编号。

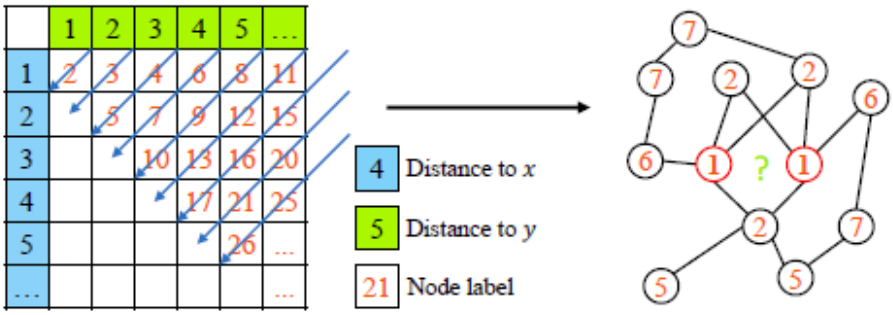
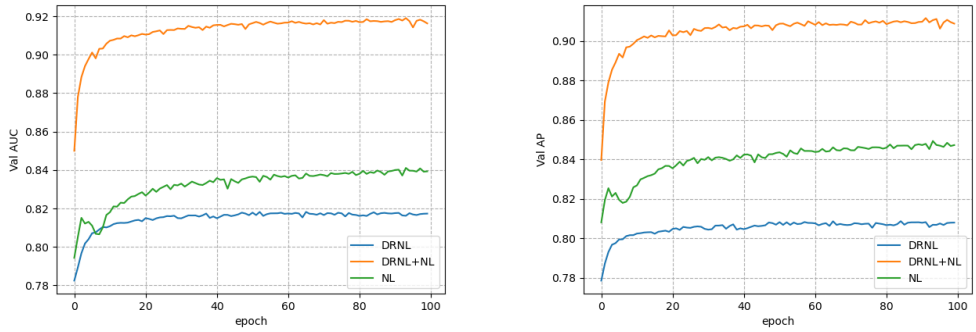


Figure 19: Double Radius Node Labeling

除了 DRNL 特征编码之外，我们还考虑其它的编码方式，考虑到对于每个节点，其本质是一个设备，具有相应的设备类别，我们将设备类别进行独热编码，对每一个节点得到一个三维的特征表示，我们将这种特征编码方式称之为 NL。同时，我们将 DRNL 编码得到的特征与 NL 编码得到的特征拼合起来，得到新的特征，称之为 DRNL+NL。本文之后的实验分别验证了这三种编码方式的合理性。

所有子图的特征信息构建完毕后，我们通过 *DGCNN* 图神经网络进行训练，注意这里是一个图的二分类任务，因此最后我们需要得到的是一个图的嵌入，在得到图的嵌入之后，我们再利用全连接层以及 *sigmoid* 函数将其转化为概率值。

在实验环境的设置上，鉴于 SEAL 的训练时间较长，我们将训练轮次设置为 100，学习率设置为 0.0001，优化器使用 Adam，并在同样的 GeForce RTX 3090 上使用 GPU 训练，使用 AUC 以及 AP 进行模型性能的度量。分别使用三种不同的初始特征编码方式进行实验，得到的结果如图20、图21、图22和图23所示。

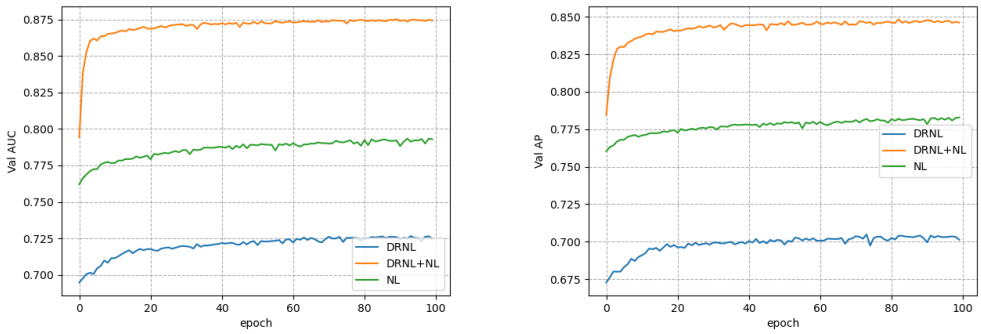


(a) AUC on validation set

(b) AP on validation set

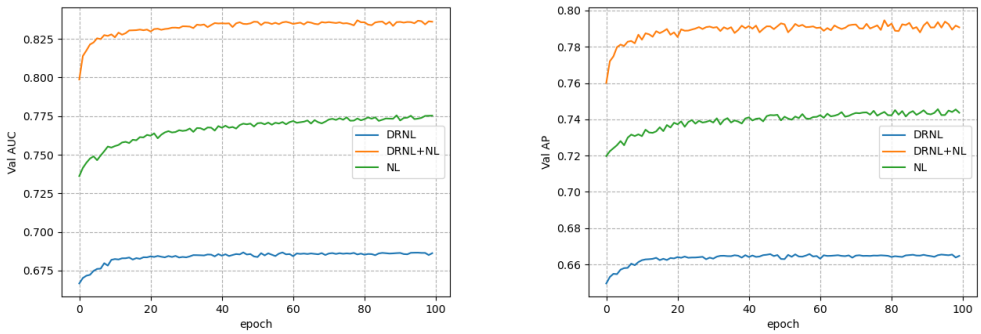
Figure 20: Metrics on the validation set(split ratio:85/5/10)

从图20和图21中可以看出，使用 DRNL 编码时其链接预测的效果明显高于使用图自编码器的结果，这说明 SEAL 利用拓扑结构的信息更好的把握住了节点之间的关系。而加入了节点类型的标签信息后，使用 DRNL+NL 这一特征编码作为初始编码时，我们可以看到，其 AUC 有非常大的提高，在测试集



(a) AUC on validation set (b) AP on validation set

Figure 21: Metrics on the validation set(split ratio:70/10/20)



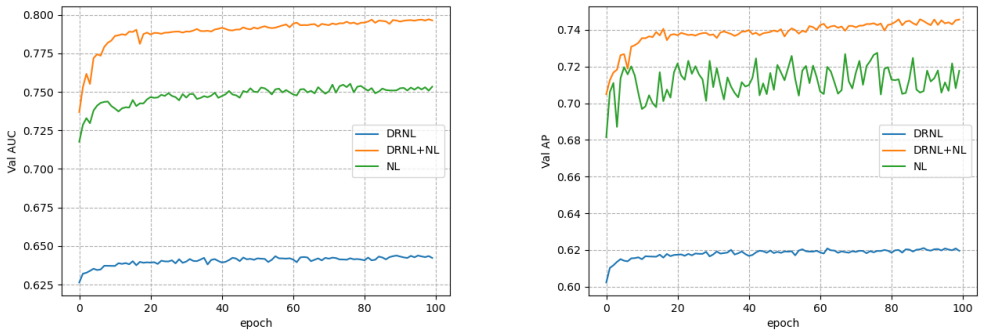
(a) AUC on validation set (b) AP on validation set

Figure 22: Metrics on the validation set(split ratio:60/10/30)

上的表现已经超过了 0.86，与仅仅使用 DRNL 相比有较大幅度的提升。同时，为了说明 DRNL 与 NL 这两种编码方式的有效性，我们也单纯使用 NL 这种编码方式进行实验，实验得到的结果如两图中绿线所示，我们看到其 AUC 也是分别逼近 84% 和 80%，但是可以看到，无论是使用 DRNL 还是 NL，其模型性能都低于二者的结合，这说明这两者都对模型起到正面的效果。

模型在验证集和测试集上的表现如表3所示，我们可以看到在三种编码方式下，验证集上的表现与测试集上的表现相差无几，这说明模型并没有过拟合。当我们降低训练集的比例时，由于原图中被当作验证集和测试集的边数增多，导致大量的边被从原网络图中删失，从而使得图的拓扑结构破坏的比较多，这样在构建训练集时，会使得构建的子图中含有的拓扑信息降低，导致训练结果有所下降。

总体来说，SEAL 的结果要高于使用 GAE 做链接预测的结果，这说明 SEAL 使用的拓扑结构的信息对于链接预测是非常重要的。论文 [28] 提出仅仅通过图神经网络去聚合节点特征来实现链接预测并不能有效地区分一些非同构的链接。而解决这个问题的比较好的方法就是使用一些 labeling trick，本文所使用的 DRNL 编码方式就是一种典型的 labeling trick，因此在模型性能上有着较大的提升。



(a) AUC on validation set

(b) AP on validation set

Figure 23: Metrics on the validation set(split ratio:50/20/30)

Table 3: Val/Test Performance of SEAL on HUAWEI Dataset

split ratio	85/5/10				70/10/20			
metrics	Val		Test		Val		Test	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DRNL	0.8009	0.7894	0.8070	0.8002	0.7246	0.7013	0.7348	0.7185
DRNL+NL	0.9142	0.9012	0.9105	0.9007	0.8743	0.8461	0.8738	0.8405
NL	0.8391	0.8409	0.8421	0.8440	0.7930	0.7829	0.8038	0.7941
split ratio	60/10/30				50/20/30			
metrics	Val		Test		Val		Test	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DRNL	0.6863	0.6647	0.6839	0.6649	0.6423	0.6196	0.6433	0.6193
DRNL+NL	0.8361	0.7907	0.8406	0.7987	0.7963	0.7456	0.7925	0.7383
NL	0.7753	0.7436	0.7771	0.7514	0.7533	0.7177	0.7531	0.7263

前文中我们也提及，电信网络的数据为典型的近似树状结构，因此，将其映射到双曲空间中是一个可以去尝试的方案，近年来双曲图神经网络的兴起，让越来越多人把双曲空间的元素引入到模型中。这里，我们将图神经网络 DGCNN 模型中最后的 MLP 层映射到双曲空间中，在双曲空间中做线性变换后再映射到欧氏空间，我们进行同上的实验，得到的结果如表4所示。

我们从表4中可以看出，当在模型中加入双曲空间的元素时，实验的结果是有所提高的，这说明我们的数据是具有双曲空间的一些性质的。与前文在 GAE 设定下使用 HGCN 作为编码器的结果作比较，我们发现，SEAL 中使用双曲元素的效果要远高于 GAE，这说明 SEAL 所构建的特征对于链接预测是非常有帮助的。

我们之前的实验是完全基于网络的拓扑结构来实现的，而且对于 SEAL 所抽取的子图来说，其特征构建的方法也相对简单，只是使用了 DRNL，DRNL+NL 以及 NL 这三种特征编码方式。但 SEAl 模型构建特征的方式是非常灵活的，在这里我们对原有的构建特征的方式做一些改进，我们通过预训

Table 4: Val/Test Performance of H-SEAL on HUAWEI Dataset

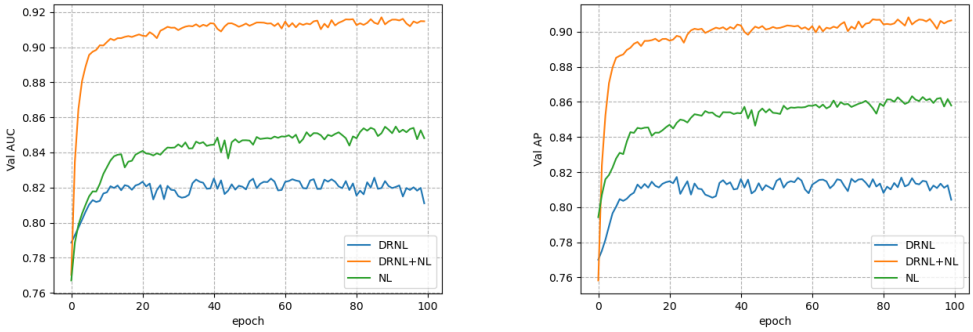
	Val AUC	Val AP	Test AUC	Test AP
DRNL	79.97	78.56	79.95	79.16
DRNL+NL	91.95	90.60	91.46	90.27
NL	85.08	84.84	84.36	84.67

练的方式来得到图中每个节点的有意义的特征表达。具体的预训练步骤如下：

- 划分训练集、验证集以及测试集作为正样本
- 通过负采样得到相应的负样本
- 将训练集中的负样本与正样本合并为新的训练集中的正样本
- 通过负采样的方式采取与现有训练集正样本数目相同的负样本
- 使用图自编码器的方法来在现有的训练图上进行链接预测

这里中间有一步是将训练集中的负样本也认为是正样本加入到训练图中，这被称为 negative injection，主要目的是为了防止训练过拟合，若不加负样本，则分类器很可能会学到关于这些正样本的个性化的信息，从而使训练出来的特征丧失泛化能力。

我们使用的以预训练的模型为 GAE，其中编码器使用 GCN，也是在前文中表现最好的模型，预训练结束后，我们将模型中的编码器部分拿出来，使用训练好的编码器来得到我们的预训练的特征，这一部分特征将作为额外的特征拼接到原始的 SEAL 的特征中去，然后我们利用新的特征进行图的二分类任务，也即是链接预测任务。链接预测任务的训练曲线如图24所示，验证集和测试集上的 AUC 和 AP 如表5所示。



(a) AUC on validation set

(b) AP on validation set

Figure 24: Metrics on the validation set(split ratio:85/5/10)

Table 5: AUC and AP on val and test set

	Val AUC	Val AP	Test AUC	Test AP
DRNL pretrained	0.8224	0.8168	0.8101	0.7953
DRNL+NL pretrained	0.9245	0.9157	0.9199	0.9082
NL pretrained	0.8580	0.8601	0.8494	0.8511

我们可以从表5中看到当额外的使用预训练的特征时，模型的表现相比于之前得到了提高，这说明预训练得到的特征还是保存了图中的部分信息，因此促进了模型性能的提升。

虽然到此为止，我们的实验已经取得了比较好的结果，但回顾一下我们的实验流程，在 SEAL 中我们只是用到了网络的拓扑信息以及图中每个节点的类别信息，值得注意的是，这是一个电信网络，图中包含着丰富的告警信息还没有被使用，因此，我们希望充分利用这些告警信息，更进一步提升模型的性能。

为了更好的使用网络中节点的告警信息，我们提出了以下的想法。我们希望利用告警信息去构建一张新的网络图，在这张新的网络图中，节点的信息保持不变，但是节点之间相连的边由它们之间的告警信息所决定。考虑到电信网络中告警的产生有根因性，因此在一个设备上产生的告警可能会引起另一个设备上产生相应的告警。这也是我们希望用告警构建一张新图的原因。鉴于在不同类型的设备上，告警会被赋予不同的名称，这给我们的分析带来了很大的不便，因此我们忽略告警的名称，只考虑告警产生的时间的影响。对于两个关联程度较高的设备，若其中的一个设备产生了一个告警，则另一个设备有较大的可能在相近的时间内也产生告警，因此对于在相近的时间内频繁产生告警的设备，我们可以认为其是相连的。我们也是鉴于此来构建新的网络结构。

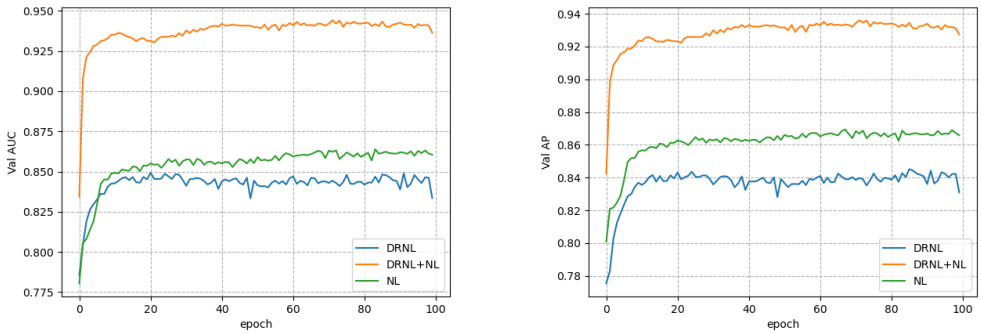
具体的如何利用告警信息来构建新的网络结构如以下步骤：

1. 忽略告警的名称，将同一个设备上所有的告警产生的时间合并到一个列表中

$$[t_1, t_2, t_3, \cdots]$$

2. 设定一个阈值 $t\_interval$，对于每一对节点，我们统计小于间隔 $t\_interval$ 的告警时间对的个数
3. 设定一个阈值 $count\_interval$,对于每一对节点,若它们在上一步统计的对数小于阈值 $count\_interval$，则我们忽略这条边，我们设立这个阈值的原因是这是一个非常大的网络结构，产生的告警的数量非常多，因此设立一个阈值是有助于我们筛除不相关的节点对的
4. 对于剩下的节点对，我们在它们之间连一条边，从而构建一张新图

在得到新构建的网络图后，由于在这个图上是没有特征的，因此我们选择用 Node2Vec 模型来生成节点的特征表示，然后该特征会被用于做 SEAl 的额外的特征输入到其中。同样的，我们划分训练集/验证集/测试集的比例为 85%/5%/10%，对于特征构建的方法，我们将无监督方式学习到的特征同样也加入其中，得到的实验结果如图25所示。



(a) AUC on validation set

(b) AP on validation set

Figure 25: Metrics on the validation set(split ratio:85/5/10)

在上述的实验中，我们设置的 $t\_interval = 300$，$count\_interval = 10$，从图25中可以看出，模型性能有了较大的提升，相比之前大概提升了 3%~4%。这说明告警中蕴含的信息是非常丰富的。

考虑到不同的阈值对于实验结果的影响，我们改变 $count\_interval$ 来多次进行实验，$t\_interval = 300$ 是不需要进行修改的，因为这个从领域专家内所得到的建议，我们设置 $count\_interval$ 从 10 到 80，间隔为 10，得到的结果如表6所示。

Table 6: Performances of different threshold

Configurations		count_threshold							
labels	metrics	10	20	30	40	50	60	70	80
DRNL	Val AUC	0.8566	0.8525	0.8391	0.8277	0.8457	0.8437	0.8515	0.8376
	Val AP	0.8496	0.8441	0.8292	0.8218	0.8370	0.8356	0.8450	0.8269
	Test AUC	0.8590	0.8555	0.8190	0.8145	0.8418	0.8426	0.8468	0.8348
	Test AP	0.8481	0.8435	0.8090	0.8045	0.8316	0.8278	0.8359	0.8208
DRNL +NL	Val AUC	0.9490	0.9380	0.9318	0.9263	0.9419	0.9451	0.9434	0.9400
	Val AP	0.9390	0.9317	0.9261	0.9188	0.9328	0.9350	0.9335	0.9306
	Test AUC	0.9450	0.9349	0.9251	0.9191	0.9411	0.9406	0.9385	0.9374
	Test AP	0.9355	0.9267	0.9164	0.9095	0.9283	0.9280	0.9262	0.9241
NL	Val AUC	0.8693	0.8608	0.8598	0.8666	0.8684	0.8482	0.8588	0.8551
	Val AP	0.8709	0.8639	0.8639	0.8688	0.8694	0.8557	0.8629	0.8598
	Test AUC	0.8582	0.8577	0.8505	0.8571	0.8633	0.8417	0.8537	0.8343
	Test AP	0.8631	0.8629	0.8534	0.8588	0.8647	0.8462	0.8553	0.8399

注意到这里的实验都只进行了一次，因此可能与真实效果存在这稍微的波动，但大体上来说，随着 *count_interval* 的增大，模型的表现呈现出下降的趋势，这说明当 *count_interval* 增加时，一些有关联的节点也被排除在外了，这在一定程度上弱化了模型的性能。

同样的，我们改变训练集/验证集/测试集的比例，分别设置为 70%/10%/20%，60%/10%/30%，得到的实验结果如表7所示。

Table 7: Val/Test Performance of SEAL on HUAWEI Dataset

	use alarm				not use alarm			
metrics	Val		Test		Val		Test	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
85/5/10	0.9490	0.9390	0.9450	0.9355	0.9142	0.9012	0.9105	0.9007
70/10/20	0.8780	0.8565	0.8834	0.8558	0.8743	0.8461	0.8738	0.8405
60/10/30	0.8270	0.7960	0.8497	0.8173	0.8361	0.7907	0.8406	0.7987

从表7中可以看出，在不同的数据集划分比例下，当使用告警时，模型在测试集上的表现要好于不适用告警数据的情况的，这再一次证实了告警拓扑图的有效性。然而，当训练集的比例下降时，模型效果的提升程度是有下降的，这是因为当训练集的比例减小时，原始的网络拓扑被破坏的比较厉害，这时候告警所提供的信息无法通过网络拓扑传播，导致性能相对下降。

我们也做了一个对比试验，通过使用 Node2Vec 训练出的特征，我们在 GAE 的实验环境中使用其作为输入数据进行训练，结果如表8所示。

Table 8: Val and Test Performance

Encoder	Val AUC	Val AP	Test AUC	Test AP
GAE(GCN)	0.6533	0.6556	0.6783	0.6774
GAE(SGC)	0.6656	0.6658	0.6628	0.6579
GAE(GAT)	0.6583	0.6486	0.6375	0.6316

从表8中可以看到，模型的表现依旧很差，这说明在缺少网络拓扑信息的基础上，构建的特征是无法起到较好的效果的。这也再一次证实了 SEAL 的有效性。

前文中我们提到过，这是一个树型的网络结构，因此，节点的类别分布是不均衡的，其中网络中节点一共分为 3 级，分别为 Routers、MicroWave、NodeB，对应的个数分别为 507，24515 和 16121，因此这是一个类别分布不均匀的数据。当我们考虑节点之间的连接关系时，由于这是一个近似树型网络结构，节点之间的边大多存在于不同类型节点之间，而我们关心的也只是不同层级的节点之间的连接关系，尤其是 NodeB 与其上层节点之间的关系，因此，对于验证集以及测试集中的边，我们只保留 NodeB 类型节点和 Routers 类型节点之间形成的边以及 NodeB 类型节点和 MicroWave 类型节点之间形成的边。我们保留训练集不变，用验证集和测试集衡量模型的训练效果，得到的结果如表9所示。

Table 9: Val and Test Performance

	Val AUC	Val AP	Test AUC	Test AP
DRNL	0.9442	0.9287	0.9422	0.9193
DRNL + NL	0.9790	0.9724	0.9747	0.9574
NL	0.9544	0.9503	0.9414	0.9291

我们看到，若只考虑 NodeB 的不同层级的边时，模型的性能表现是非常好的，这说明 SEAl 在建模这种层次化非常明显的网络结构中是非常强有力的。

到目前位置，我们利用 SEAL 模型以及提出了几种构建特征的方法，有利的提升了 SEAL 模型的性能。虽然在实际应用的过程中还有一些问题需要解决，但这个方法的效果仍然是很好的。

我们总结一下所有的实验结果，包括了主流的两种链接预测的方法 GAE 和 SEAL 以及我们基于 SEAL 模型的改进，最终的结果汇总在表10中。(train/val/test: 85/5/10%)

Table 10: Metrics of different configurations

Encoder	embedding	Val AUC	Val AP	Test AUC	Test AP
GAE(GCN)	one-hot	0.7100	0.7148	0.7126	0.7149
GAE(SGC)	one-hot	0.6912	0.7002	0.6817	0.6856
GAE(GAT)	one-hot	0.6622	0.6791	0.6751	0.6707
GAE(HGCN)	one-hot	0.6749	0.6765	0.6746	0.6784
SEAL	-	0.9142	0.9012	0.9105	0.9007
Ours	-	0.9490	0.9390	0.9450	0.9355

4 基于图粗化的大规模图学习

近年来，图神经网络（GNN）已成为图机器学习的主流工具，它在具有显式或隐式图结构的场景中有许多应用，例如社交网络、分子图、生物蛋白质-蛋白质相互作用网络、知识图和推荐系统等，实际应用中许多复杂的结构都可以用图建模。

尽管取得了巨大成功，但随着各种应用中图的规模越来越大，它们对处理、提取和分析信息构成了重大的计算挑战，将 GNN 扩展到大图的困难限制了 GNN 在大规模工业应用中的使用。

在传统的机器学习框架中，模型的损失函数可以分解为单个样本的损失之和，因此可以使用小批次处理（mini-batch）技术和随机优化技术来处理比 GPU 内存大得多的训练集。然而，GNN 的训练中，节点的表示从其邻居的表示递归计算得出，L 层 GNN 中的每个样本的损失取决于其 L 跳（L-hop）邻居诱导的子图，子图的大小随 L 呈指数级增长，使得上述一般的策略失效。因此，GNN 的训练通常采用全批次的梯度下降，这就是处理大规模图数据时所面临的问题。

4.1 大规模图学习相关方法

记号说明：考虑有权无向图 $G=(V,E,W)$，其中 V 表示顶点集，E 表示边集，W 表示边的权重（Weight）矩阵，记图 G 的邻接（Adjacency）矩阵为 A，度（Degree）矩阵为 D，拉普拉斯（Laplacian）矩阵为 $L=D-A$。

最近，大量的工作研究了在大规模图上的 GNN 训练，并提出了各种技术来提高 GNN 的可扩展性。经典的方向是解耦节点之间的相互依赖关系，从而缩小邻居消息聚合的感受域。由 [4] 开创的分层采样与小批次训练相结合的方法已经被证明是一种非常有效的策略，此后，一些后续工作试图通过优化的采样过程、更好的随机估计来改善这一基线结果。

4.1.1 Approximate personalized PageRank [2]

PPRGo 是这一思路在 PPNP 框架上的进一步应用。对于一般的 PPNP 框架，它的主要计算困难在于求解 PPR 矩阵 $\Pi=\alpha\big(I-(1-\alpha)(A+I)\big)^{-1}$，具体实现中 APPNP 采用幂法求近似解，但矩阵求幂仍然是一个巨大的开销，而且每一个批次都要重新计算一次。

PPRGo 用到了两个技巧：计算对于每一个节点的 PPR 向量，就能省去各个批次重新计算的问题，计算节点层次（point-wise）的消息传播，大大节省训练的时间；在做邻居的消息聚合时，不聚合全部的邻居信息，而只聚合 PPR 向量中权重最大的 K 个邻居节点：

$$Z_i=\mathrm{softmax}\Big(\sum_{j\in\mathcal{N}_i^k}\pi_j(i)H_j\Big)$$

其中 $\pi(i)$ 表示节点 i 的 PPR 向量，$\mathcal{N}_i^k$ 表示 $\pi(i)$ 中数值最大的 k 个节点的下标。

记 e_i 表示第 i 个位置为 1 其余位置为 0 的向量，d_v 表示节点 v 的度，$\mathcal{N}_t$ 表示节点 t 的邻域，则 PPR 向量的近似计算如算法1所示。

Algorithm 1: Approximate PPR [1]

Input: 图 G = (V, E)，传播系数 α，目标节点 v_t，允许误差范围 ε

Output: PPR 向量 $\pi^{(\varepsilon)}$

初始化 PPR 向量 $\pi^{(\varepsilon)}=0$ 和残差向量 $r=\alpha e_t$;

while $\exists v\ s.t.\ r_v>\alpha\varepsilon d_v$ **do**

  $\pi_v^{(\varepsilon)}+=\alpha r_v$;

  $r_v=(1-\alpha)\dfrac{r_v}{2}$;

  for $u\in\mathcal{N}_t$ **do**

    $r_u+=\dfrac{r_v}{d_v}$;

  end

end

4.1.2 Cluster Graph Convolutional Networks [3]

子图采样技术和上述方法比较相关，它在训练中每次迭代采样一个子图，然后简单地对这个子图执行全批次的梯度下降。由于子图规模变小，所以节省了内存开销，但在实践中，每次从一个大图执行随机采样需要对内存进行多次随机访问，这对 GPU 来说依旧是高负荷的。

Cluster-GCN 就是如此，使用聚类的思想，把图划分为小块进行训练。因为图聚类算法让相似的节点分在了一起，可能会让类内的分布和原始不一样，为了进行平衡，训练时同时抽取多个类别作为一个批次参与训练，具体的实现如算法2所示。

Algorithm 2: Cluster GCN

Input: 图 G = (V, E)，特征矩阵 X，标签 Y，迭代次数 N，图神经网络 $\mathrm{GNN_G(W)}$，损失函数 $\mathcal{L}$

Output: 节点表示 $\hat{X}$

使用图聚类算法（例如 METIS）把原图节点分成 $V = \{V_1, V_2, ..., V_c\}$ 多个部分；

for *i = 1, 2, ..., N* **do**

  不放回的随机从 V 中选择 q 个类别 $V' = \{V_{t_1}, V_{t_2}, ..., V_{t_c}\}$；

  由顶点集 V' 生成子图 G'；

  计算子图 G' 上的梯度 $g = \nabla \mathcal{L}_{G'}$ 更新 $\mathrm{GNN_G(W)}$；

end

4.1.3 Simplifying Graph Convolutional Networks [22]

SGC 代表了另一个研究方向，通过移除连续层中的非线性变换、折叠权重矩阵来减少计算过量的复杂度。从理论上分析，所得到的线性模型等价于一个固定的低通滤波器，模型对大多数下游任务没有负面影响。

基于此，图神经网络的消息传递过程可以被预先计算和存储，之后便可用简单的随机优化技巧来训练。具体来说，对于一般的 GCN 层，有更新公式（其中 $\tilde{A} = A + I$，$\tilde{D}$ 是 $\tilde{A}$ 的度矩阵）

$$H^{(k)} = \mathrm{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k-1)}\Theta^{(k)})$$

在 SGC 的更新公式中，合并所有的 GCN 层，取消激活函数，有（记 $S = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$）

$$Y = \mathrm{softmax}(S^K X\Theta^{(1)}\Theta^{(2)}\cdots\Theta^{(k)}) \triangleq \mathrm{softmax}(S^K X\Theta)$$

这样，对于固定的 k，我们只需提前与处理 S^k，模型只剩下一层，所需的训练参数只有一个权重矩阵 Θ，能够大大优化所需训练的时间。但需要注意的是，SGC 随着 k 的增大，预处理开销仍会呈指数级增大；且虽然此方法在常见的基准测试上通常比基于采样的技术性能更好，但它们只适用于 GCN 这一类基于线性变换的消息传递的图神经网络。

4.2 基于图粗化的神经网络训练

我们类比于 Cluster-GCN 的思想，把图进行简化以后在小图上进行训练。和聚类方法类似，希望寻找一些图简化（Graph reduction）的方法，缩小图的规模的同时保留关键的属性以便后续的处理和分析。

图的简化方法主要有两种。一种是减少图中边的数量，称为图稀疏化（Graph sparsification）；另一种是减少图中顶点的数量，称为图粗化（Graph coarsening）。由于没有统一的标准来衡量简化前后两个图之间的相似性，所以所有的方法都是通过特定相似度量诱导得到的。

4.2.1 Graph sparsification

图稀疏化过程是对于一个图 $G=(V,E)$，得到简化后的图 $G'=(V,E')$，即保持顶点集不变，缩小边集使得 $E'\subset E$。下面介绍三种不同的相似度度量，用来衡量稀疏化后的图 G' 和原图 G 之间的差异。

–**Spanner** [14]

图 G' 称为图 G 的 t 跨图（t-spanner of G），当且仅当满足对于任意的 $u,v\in V$，有

$$d(u,v,G')\leq td(u,v,G)$$

其中 d 表示距离函数，不同的距离函数诱导出不同的 t。

–**Cut sparsifier** [6]

对于有权图 $G=(V,E,W)$ 和集合 $S\subseteq V$，定义 S 在图 G 上的切割为

$$Cut_G(S)=\sum_{u\in S}\sum_{v\notin S}W(u,v)$$

子图 G' 称为图 G 的 ε 切割稀疏子图，当且仅当满足对于任意的 $S\subseteq V$，有

$$(1-\varepsilon)Cut_G(S)\leq Cut_{G'}(S)\leq(1+\varepsilon)Cut_G(S)$$

–**Spectral sparsifier** [18]

子图 G' 称为图 G 的 ε 谱稀疏子图，当且仅当满足对于任意的 $x\in\mathbb{R}^{\|V\|}$，有

$$(1-\varepsilon)x^TL_{G'}x\leq x^TL_Gx\leq(1+\varepsilon)x^TL_{G'}x$$

4.2.2 Graph coarsening

图粗化过程是对于一个图 $G=(V,E)$，得到简化后的图 $G'=(V',E')$，使得顶点集缩小，即 $V'\subset V$。直观来说，任何的聚类（例如谱聚类）算法都可以诱导出一种粗化。下面介绍一种相似度度量方式。

4.2.3 Restricted spectral similarity [10]

由于粗化后图的节点数减少，导致谱稀疏的类似度量不能直接使用，所以需要添加额外的约束。

图 G' 称为 $\{\varepsilon_k\}_{k=1}^K$ 受限谱相似于图 G，当且仅当存在正整数 K，使得对于任意的正整数 $k\leq K$，都存在 ε_k 满足

$$(1-\varepsilon)\lambda_k\leq x_k^TL_{G'}x_k\leq(1+\varepsilon)\lambda_k$$

其中 λ_k 表示 L_G 的第 k 大特征值，x_k 表示对应的特征向量使得 $x_k^TL_Gx_k=\lambda_k$。

4.3 算法实现 [5]

前面两个小节介绍了四种不同的理论保证，对于每一个相似性度量，都有对应的算法能够在相应的约束下完成对大图的稀疏化或粗化。

具体的实验中，我们采用图粗化的方法，如算法3所示。

Algorithm 3: Training GNN with Graph Coarsening

Input: 图 G = (V, E)，特征矩阵 X，标签 Y，图神经网络 $\mathrm{GNN_G(W)}$，损失函数 $\mathcal{L}$

Output: 网络的最优参数 W^*

使用图粗化算法（例如谱聚类粗化）把原图 G 粗化成图 G'，归一化粗化矩阵为 $\hat{P}$；

计算粗化后的特征 $X' = \hat{P}^T X$ 和标签 $Y' = \arg\max(\hat{P}^T Y)$；

计算损失函数 $\mathcal{L}(\mathrm{GNN}_{G'}(\mathrm{W}), Y')$，得到最优参数 W^*；

4.3.1 框架说明

一般的图粗化由满射 $\varphi : V \mapsto V'$ 唯一确定，称为粗化映射。对于任意的 $v'_r \in V'$，记 $\varphi^{-1}(v'_r) = \{v_i \,|\, v_i \in V \text{ 且 } \varphi(v_i) = v'_r\}$，则粗化映射还需满足使得所有 $\varphi^{-1}(v'_r)$ 对应的子图都是连通的。

粗化映射 φ 诱导的线性变换矩阵为 $P \in \mathbb{R}^{\|V'\| \times \|V\|}$，定义

$$P(r,i) = \frac{1}{|\varphi^{-1}(v'_r)|} \mathbf{1}_{\{v_i \in \varphi^{-1}(v'_r)\}}$$

其中 $|\varphi^{-1}(v'_r)|$ 表示 $\varphi^{-1}(v'_r)$ 中元素的个数。

而归一化粗化矩阵 $\hat{P}$ 定义为

$$\hat{P}(r,i) = \frac{1}{\sqrt{|\varphi^{-1}(v'_r)|}} \mathbf{1}_{\{v_i \in \varphi^{-1}(v'_r)\}}$$

4.3.2 更新公式

特别的，选取 GCN 作为我们算法使用的图神经网络，我们定义在粗化图 G' 上的修正更新式

$$Z = (\hat{P}^T D \hat{P} + I)^{-\frac{1}{2}} (\hat{P}^T A \hat{P} + I)(\hat{P}^T D \hat{P} + I)^{-\frac{1}{2}} X' W$$

注意这里的 $A_{\hat{P}} \triangleq \hat{P}^T A \hat{P}$ 并不是粗化图 G' 的邻接矩阵。

可以证明，若记 $C = \mathrm{diag}\{|\varphi^{-1}(v'_r)|\}_{r=1}^{\|V'\|}$，即 $C = (PP^T)^{-1}$，更新式等价于

$$Z = (D_{G'} + C)^{-\frac{1}{2}} (A_{G'} + C)(D_{G'} + C)^{-\frac{1}{2}} X' W$$

其中 $A_{G'}$ 是图 G' 的邻接矩阵，$D_{G'}$ 是它的度矩阵，表达式和一般的 GCN 更新式相似。

4.3.3 理论保证

对于 APPNP（GCN 是特殊情况的 APPNP）来说，记 $Z^{(t)}$ 是第 t 层的输出，则 $\{Z^{(t)}\}_{t=1}^{\infty}$ 收敛到 $Z^{(\infty)} = Z^* = \tilde{D}^{\frac{1}{2}}Y^*$，其中（$n$ 为特征的维度）

$$Y^* = \underset{Y\in\mathbb{R}^{\|V\|\times n}}{\arg\max}\,(1-\beta)\mathrm{tr}(Y^TLY) + \beta\|\tilde{D}^{\frac{1}{2}}Y - Z^{(1)}\|_F^2$$

记 L 的前 k 个特征向量构成的矩阵为 $Q \in \mathbb{R}^{\|V\|\times k}$，考虑限制在 Q 的列空间上的二次优化问题的最优值

$$Y^* = \underset{Y\in\mathrm{Col}(Q)}{\arg\max}(1-\beta)\mathrm{tr}(Y^TLY) + \beta\|\tilde{D}^{\frac{1}{2}}Y - Z^{(1)}\|_F^2$$

进一步，可以诱导出最优值 $Z'^{(\infty)}$。

可以证明，由上一节定义的公式计算的得出的序列 $\{Z'^{(t)}\}_{t=1}^{\infty}$ 收敛到 $Z'^{(\infty)}$，其中粗化矩阵 P 由基于 Q 的谱聚类粗化得出。也就是说，在谱聚类粗化后的图上做的 APPNP 训练，等价于在原图上做受限的 APPNP 训练。

这样，通过在粗化后的图 G' 上训练，能够大大减少图神经网络训练所需的参数，减少训练消耗时间和运行内存开销。

4.4 实验结果

我们在 Cora、Citeseer、Pubmed、Coauthor Physics 和 DBLP 数据集上做了关于 GCN 和 APPNP 的对比实验，其中 fixed 表示采用数据集给定的训练测试集划分，5、20 表示随机抽取每个类别各 5、20 个作为训练集，实验结果如下：

Table 11: GCN 和 APPNP 的对比实验结果图，$c = [0.7, 0.5, 0.3, 0.1]$ 表示粗化系数

	Cora		Citeseer		Pubmed		Coauthor Physics		DBLP	
	5	Fixed	5	Fixed	5	Fixed	5	20	5	20
GCN	67.5±4.8	81.5±0.6	57.3±3.7	71.1±0.7	67.4±5.6	79.0±0.6	91.2±2.1	93.7±0.6	61.5±4.8	72.6±2.3
GCN (c=0.7)	67.9±4.3	82.3±0.6	57.5±5.9	71.8±0.4	68.3±5.2	78.9±0.4	91.0±1.9	93.8±0.6	61.4±5.0	72.1±2.1
GCN (c=0.5)	68.8±4.6	82.7±0.5	57.7±5.3	72.0±0.5	68.9±4.4	78.5±0.3	91.5±2.0	93.7±0.7	61.8±4.8	72.7±2.0
GCN (c=0.3)	69.4±4.5	81.7±0.5	58.1±5.2	71.4±0.3	68.7±4.2	78.4±0.4	90.8±2.3	93.4±0.6	64.8±5.2	74.5±1.9
GCN (c=0.1)	67.6±5.1	77.8±0.7	58.3±6.3	71.1±0.4	68.5±5.2	78.3±0.5	87.8±3.6	91.5±1.4	67.9±5.6	76.0±2.1
APPNP	72.8±3.8	83.3±0.5	59.4±4.5	71.8±0.5	70.4±4.9	80.1±0.2	92.0±1.6	94.0±0.6	72.9±4.2	79.0±1.1
APPNP (c=0.7)	73.9±4.6	83.9±0.8	59.7±4.3	71.8±0.6	70.7±5.5	80.4±0.3	92.3±1.6	93.7±0.8	72.0±4.5	78.7±1.3
APPNP (c=0.5)	73.4±4.3	83.7±0.7	60.4±4.8	72.0±0.5	71.2±5.0	79.6±0.3	91.8±1.9	93.9±0.5	72.3±4.0	79.1±1.2
APPNP (c=0.3)	73.1±3.5	82.5±0.6	60.9±5.7	71.6±0.4	70.6±5.3	78.4±0.7	91.7±1.5	93.6±0.6	72.7±4.2	79.7±1.0
APPNP (c=0.1)	70.8±4.9	80.2±0.8	60.7±5.8	71.8±0.5	70.4±4.9	77.3±0.5	88.6±3.3	91.0±1.2	72.1±5.8	79.0±1.7

和 Cluster-GCN 一样，这也是可以适配所有图神经网络的的算法框架，不仅仅是基于谱聚类的粗化，下面是几种粗化方法在 Cora 和 DBLP 数据集上的对比结果：

可以看出各种粗化方法的表现差别不大，在不太大扰动模型最后呈现效果的同时，图粗化技术让我们以更快的速度完成了训练，这是令人惊喜的。

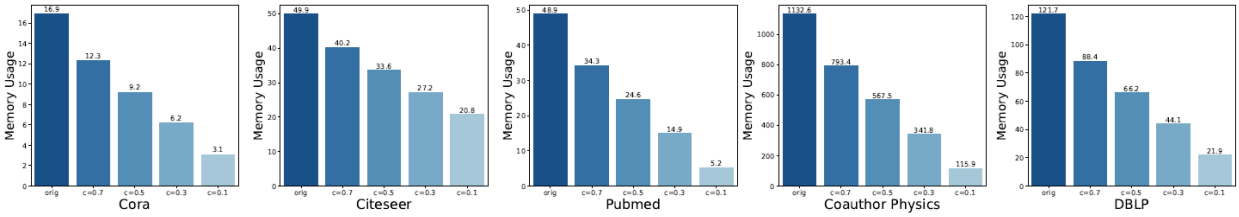


Figure 26: APPNP 的内存开销对比图

Table 12: 不同 Coarsening 方法的对比实验结果图，时间表示运行粗化算法进行数据预处理所需的时间（单位：秒）

数据集	粗化方法	$c = 0.7$			$c = 0.5$			$c = 0.3$		
		GCN	APPNP	时间	GCN	APPNP	时间	GCN	APPNP	时间
Cora	Spectral Clustering	82.2±0.5	83.2±0.4	23.4	81.5±0.7	82.5±0.5	16.3	79.4±0.5	78.0±1.3	10.0
	Variation Neighborhoods	82.3±0.6	83.9±0.8	2.0	82.7±0.5	83.7±0.7	1.3	81.7±0.5	82.5±0.6	2.1
	Variation Edges	82.3±0.5	83.6±0.6	0.3	82.2±0.5	83.9±0.5	0.5	80.0±0.4	81.1±0.7	0.6
	Algebraic JC	81.9±0.7	82.9±0.7	0.3	81.6±0.6	83.5±0.6	0.5	82.2±0.5	82.5±0.7	0.7
	Affinity GS	81.4±0.4	83.3±0.4	2.3	82.0±0.7	83.7±0.6	3.2	81.2±0.6	81.9±1.1	3.7
DBLP	Spectral Clustering	71.5±2.2	78.9±1.0	720.6	72.8±1.9	78.7±0.9	492.2	73.7±1.8	77.4±1.3	273.5
	Variation Neighborhoods	72.1±2.1	78.7±1.3	8.3	72.7±2.0	79.1±1.2	9.4	74.5±1.9	79.7±1.0	12.6
	Variation Edges	72.3±2.4	78.9±1.0	2.8	73.4±1.9	79.1±1.2	4.3	74.2±1.7	79.5±1.2	6.2
	Algebraic JC	72.5±2.3	78.6±1.6	3.0	73.1±2.0	78.3±1.1	5.5	74.0±1.7	79.1±1.2	7.3
	Affinity GS	73.2±2.1	79.2±1.6	135.7	73.9±1.7	79.6±0.7	199.6	75.3±1.6	79.9±1.1	225.9

4.5 小结

我们提出了一种用于 GNN 的可扩展训练的图粗化方法。这种方法是通用的，简单的，具有次线性训练时间和空间，并对使用粗化操作的影响进行了严格的理论分析，为粗化方法的选择提供了一定的指导。更进一步，通过理论分析表明，粗化可以被认为是一种正则化，能够提升模型的泛化性能。注意到，我们的方法和前文所提到的现有方法是不是互斥的，可以很容易地将两者结合起来以处理真正的工业规模的图。在真实数据集上的实验结果表明，简单地应用现成的粗化方法，甚至可以将节点数量减少 10 倍，而不会导致分类精度的明显降低。

5 针对于图神经网络的大规模图学习方法

在

6 总结

通过分析，我们发现拓扑还原和链路预测问题基本一致，基于此，第一阶段的工作中我们首先对对链路预测先关方法和文献的细致的调研和分析，包括：传统的基于启发式算法、基于图嵌入的方法、现有的基于神经网络以及基于子图编码的方式。进一步我们对华为的数据并进行了相关的实验，实验探索

发现，对于链路预测问题，基于子图编码的方式较其他的图表征学习方法有十分明显的优势，在华为数据集上利用子图编码以及相应的设备类别以及告警信息，可以有较好的预测效果。为了更进一步提升模型的效果，我们进一步利用了数据集中告警的信息并提供了有效的帮助。

除此之外，在工业应用中，大规模的图数据的训练问题是较为关注的问题。对于该问题，我们也进行了相应的一些探索，并提出了一种基于图粗化的高效图神经网络训练方式，我们通过理论分析，发现粗化可以被认为是一种正则化，并提升模型的泛化性能，基于图粗化的大规模图训练方式具有很好的通用性。

References

[1] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486, 2006.

[2] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. *Scaling Graph Neural Networks with Approximate PageRank*, page 2464–2473. Association for Computing Machinery, New York, NY, USA, 2020.

[3] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 257–266, New York, NY, USA, 2019. Association for Computing Machinery.

[4] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

[5] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling Up Graph Neural Networks Via Graph Coarsening. *arXiv e-prints*, page arXiv:2106.05150, June 2021.

[6] David R. Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, page 648–657, New York, NY, USA, 1994. Association for Computing Machinery.

[7] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016.

[8] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[9] Zhiyuan Liu and Jie Zhou. Introduction to Graph Neural Networks. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(2):1–127, 3 2020.

[10] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3237–3246. PMLR, 10–15 Jul 2018.

[11] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, 9 2013. arXiv: 1301.3781.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *arXiv:1310.4546 [cs, stat]*, 10 2013. arXiv: 1310.4546.

[14] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online Learning of Social Representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 8 2014. arXiv: 1403.6652.

[16] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467, 2 2018. arXiv: 1710.02971.

[17] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 1 2009. Conference Name: IEEE Transactions on Neural Networks.

[18] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[19] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale Information Network Embedding. *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 5 2015. arXiv: 1503.03578.

[20] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile Graph Embeddings from Similarity Measures. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, pages 539–548, 2018. arXiv: 1803.04742.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[22] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 09–15 Jun 2019.

[23] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr. au2, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks, 2019.

[24] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[25] Yuan Yin and Zhewei Wei. Scalable Graph Embeddings via Sparse Transpose Proximities. *arXiv:1905.07245 [cs, stat]*, 5 2019. arXiv: 1905.07245.

[26] Muhan Zhang and Yixin Chen. Weisfeiler-Lehman Neural Machine for Link Prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 575–583, Halifax NS Canada, 8 2017. ACM.

[27] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems*, 31:5165–5175, 2018.

[28] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Revisiting graph neural networks for link prediction, 2021.

[29] Tao Zhou, Linyuan Lu, and Yi-Cheng Zhang. Predicting Missing Links via Local Information. *The European Physical Journal B*, 71(4):623–630, 10 2009. arXiv: 0901.0553.