# 基于深度聚类的
# 大规模图中的节点特征增强

*

（*　指导教师：*）

[摘要]：图结构广泛运用于各类场景如论文引用关系网等，这些场景中主要包含三类任务：节点级别任务、边级别任务以及图级别任务，其分别对节点、边和整个图进行分类和属性预测等。作为图中最基础的元素，节点在以上所有任务中都有重要的研究价值，如何对其特征进行有效的表征始终是一个重要课题。许多工作对某个节点进行表征时主要单独考虑其在图中的信息，而缺乏从节点集的角度进行节点表达的研究，但是直观上，节点集对其中的节点蕴含着相比于图中信息更加细致与独特的信息，这或许有助与提高节点表达。为此，有工作使用图分割将相似的节点归为一类，然后进行进一步的节点表达，然而它们通常是两步的方法，即先使用训练好的模型编码得到嵌入后再使用图分割算法得到节点集，之后进一步编码解码，这种方法将特征编码和分割解耦使得分割算法不是针对特定任务的。为此，本文提出了一种即插即用的基于聚类的插件式模型以及其训练与测试流程。本算法首先使用图分割算法获得初始划分，为每个簇创建簇节点以及相应边并将它们加入原图获得增广图。将原模型整合到本模型中后可以编码增广图便能获得被增强的节点表征以及更新后的图划分。本模型将节点特征编码以及图分割整合到一起，能够面对不同任务同时进行表征学习以及分割学习，同时得益于插件形式，本模型能适用于多数现有模型，增强它们的节点表达效果从而一定程度提高相应任务的指标。

[关键词]：图神经网络；节点级任务；大规模图；图分割

[**ABSTRACT**]: Graph structures are widely used in various scenarios such as thesis citation networks, which contain three main types of tasks: node-level, edge-level and graph-level tasks, which classify and predict attributes of nodes, edges and the whole graph, respectively. As the most fundamental element of graphs, nodes have important value in all above tasks, and how to characterize them effectively is always an important topic. Many works characterize a node mainly by considering its information in the graph alone, but there is a lack of research on node representation from the perspective of node sets, but intuitively, node sets contain more detailed and unique information about the nodes than that in the graph, which may help to improve node representation. To this end, there are works that use graph partition to group similar nodes together and then perform further node representation, however, they are usually two-step approaches, i.e., first using a trained model to get nodes' embedding and then using a graph partition algorithm to obtain the node set, followed by further encoding and decoding, which means the partition algorithm is not task-specific. Therefore, this paper proposes a plug-and-play plug-in model based on clustering and its training and testing process. The algorithm first uses a graph partitioning algorithm to obtain an initial partition, creates cluster nodes and corresponding edges for each cluster and adds them to the original graph to obtain an augmented graph. After integrating the original model into this model, the augmented graph can be encoded to obtain the enhanced node representations and the updated graph partitioning. This model integrates node feature encoding and graph partitioning into one model, which can perform both representation learning and partitioning learning for different tasks, and thanks to the plug-in format, this model can be applied to most existing models to enhance their node representation effects and thus improve the metrics of the corresponding tasks to a certain extent.

[**Key words**]: GNN; Node-level tasks; Large-scale graphs; Graph partition

# 目录

# 1. 引言

## 1.1 图结构

本文研究图为图论中的图结构而非人们通常所说的图片，以下将从数据结构方面描述图像（以及类似的数据如文本）和图结构的特定与区别。

日常生活中最常见到的数据是图像，文本和音频。直观来看，图像是由许多像素点构成的网格结构，文本和音频分别是一个个字符或者信号值排列形成得序列结构。如果统一称每个像素、字符和信号值为节点，那么以上数据中地节点都排列整齐，节点的邻居数量和位置是有规律的甚至是固定的，所以可以直接使用张量进行显式地表达。而节点之间的距离也很好定义，通常是欧几里德距离（欧氏距离）。假设节点的特征变量为 $X$，其维度为 $h$，那么两个节点 $v_i, v_j$ 之间的欧式距离 $d$ 定义为

$$d = \sqrt{(x_{i,0} - x_{j,0})^2 + (x_{i,1} - x_{j,1})^2 + \ldots + (x_{i,h} - x_{j,h})^2}$$

由于以上性质，图片、文本和音频等数据称为欧几里德结构数据。

然而图是非欧几里德结构数据，其性质与以上数据有着显著的差异。首先图是由顶点和连接顶点的边构成的，图可以表示为 $G = (V, E)$，其中 $G$ 代表图、$V$ 表示顶点集、$E$ 表示边集。不同于图像等数据，每个顶点的邻居 $N(V)$ 数量并不一致同时它们的位置也难以定义，所以顶点之间的距离并没那么显然，距离可能根据顶点和边的性质进行定义。此外，图的存储方式与欧几里德数据结构有所不同，其需要两个而不是一个张量存储，分别用来存储顶点的特征变量以及图的邻接矩阵（或者邻接链表）。令图 $G$ 中的顶点数量为 $|V|$，那么邻接矩阵 $M$ 大小为 $|V|x|V|$，一般来说若 $m_{ij}$ 的值非 0 则表示存在从顶点 $i$ 到顶点 $j$ 的边。虽然图片、文本等是排列整齐，但也能转换成非欧几里德结构，用图的形式表达——将像素看为顶点，相邻的像素之间即有连边，这样可以用更加抽象和一般的方式处理常见的数据。另外，图也有很多类型：按照边是否有方向分为无向图和有向图；按照顶点和边类型的多样性区分分为同构图和异构图，同构图中每个顶点属于同一个类型、每个边也属于同一个类型，比如三个论文相互引用，其中每个顶点是“论文”类型，每个边是“引用”类型，异构图存在多个类别的顶点和边，比如一个用户分别买和卖了商品 1 和商品 2，其中顶点分为“用户”和“商品”而边分为“买”与“卖”。本文只考虑无向图和同构图，实际实现中如果是有向图则将其转为无向的。

之后章节所说的“图”都默认为图结构而不是图像或影像。

## 1.2 图任务与图神经网络

以研究目标层次分类，与图有关的任务可以分为三个：节点级别任务、边级别任务和图级别任务。节点级别任务包括对节点属性进行预测和对节点分类等，研究的目标就是节点本身，如果边上有特征那么通常也会将边特征整合到节点中再进行讨论，具体任务有通过论文间的引用关系预测某个论文所属的学科；边级别任务包括通过

节点预测边的属性，研究目标是边但也需要考虑节点特征，具体的任务有给定一些属性不同的蛋白质来预测它们之间的联系；图级别任务关注整个图的性质，其先编码节点和边的特征然后整合这些信息进一步获得图的表征，具体的任务包括通过一个分子中原子的属性以及原子之间的关联来预测该分子的类别或者属性。无论哪种任务，节点作为图中最基础的元素都必须进行编码和表达，是任何模型和算法必须考虑的一环，节点特征学习的好坏以及与任务的契合度都会深刻影响最终结果，因此节点的表征学习十分重要。为此，本文着重研究节点本身的表征学习以及与其联系紧密的节点级任务。

由于图结构的特殊性，深度学习领域中传统的卷积操作不适用于图，因为卷积需要有固定的邻居数量而图中每个顶点的邻居数不定，简单地套用卷积等常用的深度学习操作不仅会导致低效的计算效率和冗余的空间资源消耗更有可能因为不合适的算法使得下游任务无法完成。为此，出现了基于消息传递范式的一系列如 GCN[1]等图神经网络算法，其主要思想是将邻居顶点信息聚合到目标顶点，包括了邻居顶点信息变换、邻居顶点信息聚合以及聚合后的目标顶点信息变换三步。此外，这些算法还分为直推式和归纳式，直推式模型在训练阶段需要输入测试数据，所能处理的图结构是固定的，而归纳式模型训练时只需要训练数据能够随时处理新加入的顶点与边。为了更具有通用性，本文设计了插件形式的归纳式模型，其也能适用于直推式模型。

## 1.3 大规模图与图分割

各类任务的图规模不一样，图级别任务的样本以图为单位，每个图的规模小，顶点数量从几十到几百不等，边的数量一般最多也就几千；然而节点级别和边级别以节点和边为单位，一般只有一个图，图规模大，尤其是节点级别任务，顶点数量可达上百万而边数量可以达到千万甚至亿级别。这种大规模图造成的内存消耗问题、模型运行速度问题始终困扰着研究人员，如何使算法模型具备可扩展性是当下研究热点。为此，很多工作致力于降低算法空间和时间上的复杂度以缓解大规模图带来的问题。部分工作利用巧妙的数学公式推导与优化算法复杂度，以坚实的理论基础和准确的代码实现取得不错的效果。

此外也有不少工作从工程的角度，通过优化采样或者图分割的方式，缓解大规模图的资源问题，平衡算法的效率和效果。在优化采样方面主要有三种角度，分别限制每个顶点、模型每一层或者模型整体采样的邻居数量，因此一个大图可以只采样部分顶点从而缓解大规模图的问题。图分割的方式将大图划分成几个互不重叠的子图，然后分批进行训练。总的来说，这些方法的本质都是用子图的数据分布近似整个图的数据分布。

图分割除了可以用于减小大规模图的压力，其很多时候用于图聚类，具有相似特征的节点会被分割到同一簇中。由于簇包含相同顶点的信息，所以一些工作将顶点用现有模型编码得到隐藏变量后使用图分割算法获得簇，然后利用这种信息对顶点进行进一步编码解码，然而它们大多数是多步的，编码的模型与图分割（聚类）模型未能很好融合，导致聚类效果是非任务导向的，不是针对特定任务的。实际上，现有工

作对顶点的编码还是主要研究单个顶点在图中的信息，以顶点集（如簇）为单位对顶点进行表征的研究还是较少。

### 1.4 本文贡献

本文提出了 ClusterALL 算法，具体贡献为：

- 通过对原大图进行图分割并加入可学习的簇中心节点，使得在小批量学习时，每个样本可以从簇中心节点获取跨批的非局部信息，解决了大规模图中小批量学习时每个样本只能获取当前批信息的问题。
- 添加的可学习的簇中心节点能够高度抽象出一类样本的表征，只要样本在本算法抽象出的空间内相似便能连接起来，此时它们可能在全图中任意位置，这种特性可以使得样本具备全图感受野，能够增强节点表征效果。
- ClusterALL 为插件式模型，本文创建了接入该模型的接口，使得大部分现有模型能够轻易套用在本模型中实现即插即用。
- 借助插件式的特性以及深度聚类，相比于原模型编码的表征，本算法能够获得更加适用于聚类的节点表征，并得到针对不同现有模型以及不同节点级节点分类任务的聚类结果。

## 2. 相关工作

### 2.1 节点表征学习

节点作为图中的基础元素，节点表征学习在几乎所有任务中都要进行重点讨论，为此也有很多工作从不同角度来优化学习效果。一些工作如 Deepwalk[2]、LINE[3]、node2vec[4]、SDNE[5]和 Struc2Vec[6]等在原图输入其他模型进行编码解码之前对原图进行操作来增强节点的特征，从而提高其他模型在不同任务中的效果。它们通常使用各自的策略对节点邻居进行采样，然后研究相邻节点的共现关系，以此学习节点的表征。这种表征学习策略大多数时候是在节点的邻域中进行，学习的信息是局部缺乏全局感知，可能会遗失图中更有效的信息。有些工作如 FLAG[7]等使用对抗数据增强的策略，向原特征空间中添加微小的扰动来改善节点的原特征以提高现有模型的鲁棒性以及泛化能力。以上工作可以算是数据的预处理，与特定模型耦合度不高可以较为轻松地插入其他模型进行使用，不过对于不同图它们需要格外的训练，这种预处理的方式也通常与任务无关不能进行任务导向的特征增强。

实际上所有图神经网络都在进行隐式的节点表征学习，经典的图神经网络如 GCN[1]、GraphSAGE[8]、GAT[9]基于消息传递范式，其通过堆叠多层将多跳邻居节点的特征传递与聚合到目标节点中，然后根据不同的下游任务使用对应解码器解码得到输出，每一层的主要步骤分为以下两部分：

$$\alpha_i^l = \Theta(\{h_j^{(l-1)} : j \in \mathcal{N}(v_i)\}), \quad h_i^l = \Phi(h_i^{l-1}, \alpha_i^l)$$

节点 $v_i$ 在 $l-1$ 层使用函数 $\Theta(\cdot)$ 结合若干跳邻居 $\mathcal{N}(v_i)$ 的表征得到 $\alpha_i^l$，然后通过函数 $\Phi(\cdot)$ 将自身特征和邻居特征结合得到下一层即 $l$ 层的特征。基于该范式，该领域出现了两个方向：基于谱分解的方法和基于空间结构的方法。谱方法如 GCN[1]等基于图信号处理，在谱域进行卷积计算，其虽然具有坚实的理论基础但几乎都是直推式，无法处理新加入的节点故实际应用有限。基于空间结构的方法如 GAT[9]和 GraphSAGE[8]等可以归纳式学习，能够应对各种未知节点。然而大多数模型都是提取局部特征，对于某个节点，全图信息的使用率低且信息利用效果不理想。

Belkin[10]、Dwivedi[11]、Haorui[12]、Yunsheng Shi[13]等人的工作向节点特征空间中添加位置编码或者标签编码来增强节点携带的信息，以此提高模型最终的效果。这些工作也类似第一种预处理形式的方法，不过算法规模和运行成本相比来说更小，有时是作为锦上添花的技巧，本文不考虑这种形式的特征增强方法。

## 2.2 感受野和可扩展性

因为节点级任务通常使用包含上千节点甚至百万节点大规模图，其中边数量更是能达到千万甚至亿级别，所以直接读取和操作整个图会消耗大量资源，而很多时候对全图的操作是不可行的，这也是为什么图神经网络大多使用局部信息。但是局部信息始终不全面，许多工作仍然希望获得全局的信息来增强模型效果。为此，节点级模型主要从两个角度来平衡模型感受野和可扩展性，即适应图规模快速增长的能力：1. 优化模型复杂度 2. 使用采样。

Nodeformer[14]是第一个角度的典型，其将 Transformer[15]应用于大规模图中。Transformer[15]自提出以来，得益于其注意力机制，它拥有强大的全局信息整合与表达能力。令输入数据的特征向量是 $H$，$W_q, W_k, W_v$ 是将特征向量分别投影为“查询”，“键”和“值”的矩阵，$\Psi$ 为正规化函数，那么注意力机制 $attn$ 可以表示为

$$\mathcal{A} = softmax(\Psi((HW_q)(HW_k)))HW_v$$

然而该算法关于样本数量的时间和空间复杂度为 $O(n^2)$，虽然 Graphormer[16]等模型使用该架构在图级别任务中获得成功，但其无法应用于节点级大规模图中。很多工作如 Informer[17]，Autoformer[18]和 Performer[19]优化了 Transformer[15]的复杂度但尚未针对图领域，而 Nodeformer[14]充分利用了图结构且将复杂度降至线性使之具有可扩展性。

也有许多研究利用采样来缓解可扩展性的问题，其通常会使用各种采样方法从大图中获取子图进行特征编码，采样方法大致分为三种：1. 节点级采样：GraphSAGE[8]和 PinSAGE[20]等算法对每个节点采样的邻居数量进行了限制而不是采样所有邻居 2. 层级采样：FastGCN[21]等算法限制模型中每一层的邻居采样数量 3. 子图级采样：ClusterGCN[22]和 GraphSAINT[23]等工作不同与前两者围绕一些中心节点对邻居采样，其在整个大图中使用某种策略直接采样若干个子图。这些方法提高了模型感受野的同时，内存和时间消耗的增长也不会太大。

## 2.3 图分割与聚类

图分割可以用于子图采样如 ClusterGCN[22]，也能用于聚类。传统的图分割算法如 Metis[24]和 Graclus[25]等直接利用点和边特征通过贪心算法能在大规模图中获取较好的分割，基于深度学习的算法如 DAEG[26]和 GAP[27]等通常会对节点和边进行编码之后再进行分割与聚类。由于聚类通常会将拥有相似特征的边和节点放入同一个簇中，所以直观上簇包含能概括本簇中所有节点和边的特征信息，这种信息有利于节点或者边的编码。然而现有图分割算法只针对聚类任务，没有探究聚类在节点特征编码方面的作用，针对除聚类任务之外的下游任务如节点级任务较少。许多工作通常是先获得聚类后再进行进一步编码解码来得到结果，聚类结果在这种两步式流程中由于不是任务导向的可能并不能很好地适应特定任务。虽然 ClusterGCN[22]中使用了图分割获得划分并以此来训练，但其没有显式地编码与利用每个簇的特征。

# 3. 问题定义

不失一般性地，令 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 表示为一个拥有上万节点的大规模图，该图拥有节点数为 $V$ 的节点集 $\mathcal{V}$ 和边数为 $E$ 的边集 $\mathcal{E}$。边集通常会表示为一个邻接矩阵 $\mathbf{A} \in \mathbb{R}^{V \times V}$，若存在边 $(v_i, v_j), 0 \leq i, j \leq V$，那么邻接矩阵 $\mathbf{A}$ 中的元素 $a_{ij}$ 为 1，若不存在则为 0。对于任意一个节点 $v$，其拥有特征 $x_v \in \mathbb{R}^{D_{node}}$；对于任意一个边 $e$，其拥有特征 $x_e \in \mathbb{R}^{D_{edge}}$。本文针对节点级的节点分类任务，每个节点 $v$ 拥有一个标签 $y_v \in \mathbb{R}^O$，那么节点分类任务可以表示为：

$$\mathbb{F}(X_{node}, X_{edge}, \mathbf{A}) \to \tilde{Y}$$

其中 $\mathbb{F}$ 为模型如 GCN 等，$X_{node} = \{x_v | v \in \mathcal{V}\}$ 为节点特征集合，$X_{edge} = \{x_e | e \in \mathcal{E}\}$ 为边特征集合而 $\tilde{Y}$ 为模型得到的节点分类结果。此类任务致力于提高分类准确度即

$$max \quad \frac{1}{V} \sum_{v \in \mathcal{V}} \mathbb{I}(\tilde{y_v} = y_v)$$

其中 $\mathbb{I}$ 为指示函数，当其中条件成立时为 1，反之为 0。

一般来说模型 $\mathbb{F}$ 可以分为直推式和归纳式，直推式模型在训练时能够看到测试数据但是只能处理固定的图结构无法分类加入的节点，而归纳式模型训练时只需要训练数据，可以分类从未见过的节点。为了更具实用性，本文算法设计为归纳式。

# 4. 提出的算法：ClusterALL

受启发于 Graphormer[16]等研究中利用“读出”操作获取全图表征的做法以及 ClusterGCN[22]的聚类预处理，本文提出了基于聚类的节点表征增强算法 ClusterALL。该算法的核心思想是通过图分割获得图聚类，然后向原图中添加簇中心节点显式编码与“读出”簇特征，最终聚合簇中心节点和节点来增强节点表征，从而提高下游任务的效果。本算法包含了数据预处理模块、表征增强模块以及一系列训练与推断流

程，主要针对节点级任务，旨在利用可学习的聚类以及簇中心节点在全图的感受野捕获有助于增强各个节点表征的有效信息，从而提高不同模型在相关任务中的效果。此节将先提出本算法 ClusterALL 的整体流程包括算法的输入输出以及其训练与测试过程，之后进一步详细阐述算法的核心模块——预处理模块和表征增强模块，然后对本算法从聚类角度解释并提出相应的模型训练策略，最后进行算法的复杂度分析。

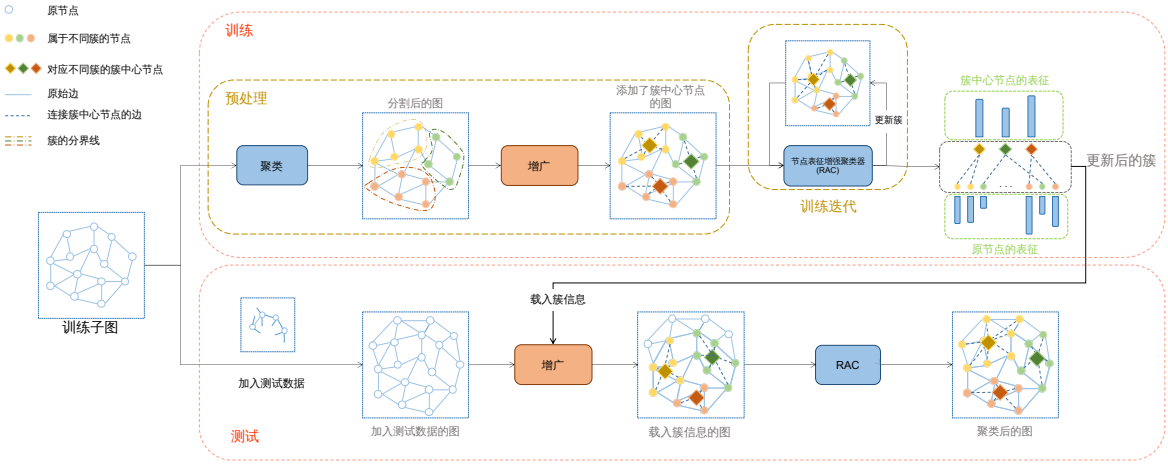


**图 1 ClusterALL 算法流程**

## 4.1 算法流程

设原图 $\mathcal{G}=(\mathcal{V},\mathcal{E})$ 含有 $V$ 个节点和 $E$ 条边，且原图可以分为两个互不相交的子图 $\mathcal{G}=\{\mathcal{G}_{train},\mathcal{G}_{test}\}$，其中 $\mathcal{G}_{train}=(\mathcal{V}_{train},\mathcal{E}_{train})$ 为包含 $V_{train}$ 个节点和 $E_{train}$ 条边的训练子图，$\mathcal{G}_{test}=(\mathcal{V}_{test},\mathcal{E}_{test})$ 为包含 $V_{test}$ 个节点和 $E_{test}$ 条边的测试（包含验证集）子图。两个子图的节点集合互斥，边集合同理，故 $V=V_{train}+V_{test}, E=E_{train}+E_{test}$。原图的节点特征为 $X=\{X_{train},X_{test}\}\in\mathbb{R}^{V\times D}$，其中 $X_{train}\in\mathbb{R}^{V_{train}\times D}$ 和 $X_{test}\in\mathbb{R}^{V_{test}\times D}$，$D$ 为特征维度。

ClusterALL 算法流程见图1（放大版的图见附录中图18），本算法包含一个核心概念和两个核心模块：

- 核心概念：簇中心节点（cluster center node），记为 CCnode。簇中心节点在图分割后加入原图，用来代表和表征一个簇，簇中心节点会被注册入核心模块二中的 RAC 模型中进行学习。令图分割获得 $k$ 个簇，则簇中心节点集 $\dot{\mathcal{V}}$ 包含 $k$ 个簇中心节点，簇中心节点的特征矩阵为 $\dot{X}\in\mathbb{R}^{k\times D}$。下文中带有 $\cdot$ 上标的字符均代表与簇中心节点有关。

- 核心模块一：预处理模块 $\mathbb{C}$。预处理模块中包含两个主要步骤：图分割操作 $\mathbb{P}$ 和图增广操作 $\mathbb{A}$。令聚类数量为 $k$，图分割操作将原图的节点分为多个簇（节点集）$\{\mathcal{P}_i\}_{i=1}^{k}$，这些簇为原节点集 $\mathcal{V}$ 的划分。图增广操作将簇中心节点连接到原图，那么增广后的图为 $\check{\mathcal{G}}=(\check{\mathcal{V}},\check{\mathcal{E}})$，其中增广后的节点集 $\check{\mathcal{V}}$ 包含 $\check{V}=V+k$ 个节

点，增广后的边集 $\check{\mathcal{E}}$ 包含 $\check{E} = E + k$ 个边。增广后的特征矩阵为 $\check{X} \in \mathbb{R}^{(V+k)\times D}$。下文中带有 $\vee$ 上标的字符表示其已被增广。

- 核心模块二：表征增强模块 $\mathbb{F}$。其中包含一个深度学习模型表征增强聚类器 RAC，记为 $\mathbb{M}$ 和一个簇更新操作 $\mathbb{U}$。

以上模块和概念将在后面小节详细说明。在算法流程示意图中，图结构由多个圆圈和边组成并被矩形框包围着，图结构中各个形状与图标的意义可见图1左上角的图例：空心圆圈代表为被聚类的节点；实心圆圈是已经聚类之后的节点，不同的颜色表示属于不同的簇；实心矩形代表簇中心节点，其颜色对应不同簇；实线是原图中的边；虚线代表节点与簇中心节点的连边；不同颜色的点划线是不同簇的边界线。

整个流程从训练子图 $\mathcal{G}_{train}$ 开始，算法的训练和测试部分略有不同：训练部分会学习模型参数以及簇并得到优化后的聚类结果；测试时会将测试子图 $\mathcal{G}_{test}$ 和优化后的聚类结果载入训练子图 $\mathcal{G}_{train}$ 进行推断，最后测试节点也会被归入到相应簇中。

#### 4.1.1 训练流程

**Algorithm 1:** Training Step

**Data:** training subgraph $\mathcal{G}_{train} = \{\mathcal{V}_{train}, \mathcal{E}_{train}\}$, node features $X_{train}$, label $Y_{train}$, number of clusters $k$

**Result:** node representation (including cluster nodes) $\tilde{X}_{train}$, clusters $\{\mathcal{P}_i\}_{i=1}^{k}$, RAC's parameters $\{W\}_{l=1}^{5}$

pre-processing $\check{\mathcal{G}}_{train}, \check{X}_{train}, \{\mathcal{P}_i\}_{i=1}^{k} \leftarrow \mathbb{C}(\mathcal{G}_{train}, X_{train})$ ;

Output $\tilde{X}_{train}, \{\mathcal{P}_i\}_{i=1}^{k}, \{W\}_{l=1}^{5} \leftarrow \Gamma(\mathbb{F}, \check{\mathcal{G}}_{train}, \check{X}_{train}, Y_{train})$ ;

训练时将训练子图 $\mathcal{G}_{train}$ 输入，经过预处理模块 $\mathbb{C}$ 和表征增强模块 $\mathbb{F}$ 的迭代训练后可以获得模型 $\mathbb{M}$ 学习好的参数 $\{W_l\}_{l=1}^{5}$、训练节点的表征以及更新后的聚类结果即每个训练顶点对应的簇。

预处理模块 $\mathbb{C}$ 中包含聚类操作 $\mathbb{P}$ 和图增广操作 $\mathbb{A}$（如图1中 pre-processing 部分所示），聚类操作使用 Metis 算法作为聚类器，其可以将图进行划分获得 $k$ 个互不重叠的簇，此聚类结果将作为初始状态为后续的操作进行铺垫。得到初始的簇之后将进行图增广 $\mathbb{A}$，为每个簇创建一个簇中心节点并加入训练图 $\mathcal{G}_{train}$ 中，然后各个簇中的节点连接它们相应的簇中心节点即添加额外的边。总的来说，预处理模块 $\mathbb{C}$ 将输入的图 $\mathcal{G}_{train}$ 聚类并添加了多个簇中心节点和边，输出增广图 $\check{\mathcal{G}}_{train}$、被增广的节点特征 $\check{X}_{train}$ 以及聚类结果。

增广之后的图便可以输入表征增强模块 $\mathbb{F}$ 中进行模型训练以及聚类的更新，此为图1中的训练迭代部分 $\Gamma$，在每一轮迭代中，被训练的图的聚类结果都将得到更新，当到达迭代最大值后将输出最终的聚类结果，该聚类结果与所用模型和当前任务密切相关。

训练流程的公式化表达见算法1，其中预处理模块 $\mathbb{C}$ 和表征增强模块 $\mathbb{F}$ 与相应的模型训练策略 $\Gamma$ 将在小节4.2和小节4.3详细阐述。

### 4.1.2 测试推理流程

测试过程需要输入训练子图 $\mathcal{G}_{train}$、测试子图 $\mathcal{G}_{test}$ 以及训练所得簇，将它们整合好之后输入表征增强模块 $\mathbb{F}$ 即可获得测试节点所对应的簇以及测试节点的表征。具体来说，测试子图 $\mathcal{G}_{test}$ 先与训练子图连接 $\mathcal{G}_{train}$ 获得图 $\mathcal{G}'$（大多数情况下图 $\mathcal{G}'$ 即为初始大图 $\mathcal{G}$），然后加载训练获得的聚类结果并加入簇中心节点以及相应的与训练节点连接的边进行增广。将包含簇信息的图输入训练好的表征增强模块中运行一次即可将测试节点归纳到不同簇之中同时获得它们的表征，此时聚类结果也将得到更新。公式化的测试流程见算法2。

以上算法流程显示地设置了全局的簇中心节点，在训练过程中簇中心节点的特征得到持续的学习并保存在模型中，模型也会学习如何利用簇中心节点的信息编码节点与增强其特征表达，同时簇也会进行更新以适应节点的编码。在测试阶段，训练数据也会参与推断，为测试数据的聚类提供一定程度的监督信息。

**Algorithm 2:** Inference Step

**Data:** training subgraph $\mathcal{G}_{train}$, testing subgraph $\mathcal{G}_{test}$, node features $X = \{X_{train}, X_{test}\}$, clusters $\{\mathcal{P}_i\}_{i=1}^{k}$

**Result:** node representation (including cluster nodes) $\tilde{X}$, prediction $\tilde{Y}$, clusters $\{\mathcal{P}_i\}_{i=1}^{k}$

1 adding testing graph into training graph $\mathcal{G}' \leftarrow \mathcal{G}_{train} \bigcup \mathcal{G}_{test}$ ;

2 augment graph with trained clusters $\check{\mathcal{G}}', \check{X} \leftarrow \mathbb{A}(\mathcal{G}', X, \{\mathcal{P}_i\}_{i=1}^{k})$ ;

3 Output $\tilde{X}, \tilde{Y}, \{\mathcal{P}_i\}_{i=1}^{k} \leftarrow \mathbb{M}(\check{\mathcal{G}}', \check{X})$ ;

## 4.2 预处理模块

预处理模块 $\mathbb{C}$ 包含图聚类操作 $\mathbb{P}$ 和图增广操作 $\mathbb{A}$，如图1中 pre-processing 中所展示。本文的图聚类策略是先使用一个快速的不需要学习的算法对原图进行初始化划分，然后通过本模型的训练来优化聚类结果。这类图聚类或分割算法有 Louvain[28]、Graclus[25]和 Metis[24]算法等，它们的划分目标是最大化簇内节点之间的相似度且最小化簇之间的相似度，评价该相似度的方法不同算法不一，如 Metis 尽可能使簇内边多而使簇间边少。本文参考 ClusterGCN 选择了多层次 k 路划分实现的 Metis 算法，其有多个优点：1. 运行效率高，百万级别的大图只需要十几秒就能划分成上百个簇，比其他常见的图划分算法快一到两个数量级，非常适合于大规模图，具有可扩展性 2. 将图划分为若干个节点数量较均等的簇，增加每个簇初始的感受野，使得簇的在学习初期能够获得更多信息进行调整。簇的数量 k 作为超参数需要人为设置，聚类结果由以下过程得到

$$\{\mathcal{P}_i\}_{i=1}^{k} = \mathbb{P}(\mathcal{G}_{train})$$

其中图聚类操作 $\mathbb{P}$ 为 Metis 算法。

**Algorithm 3:** Graph Augmentation $\mathbb{A}$

**Data:** graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, node features $X$, clusters $\{\mathcal{P}_i\}_{i=1}^{k}$
**Result:** augmented graph $\check{\mathcal{G}} = \{\check{\mathcal{V}}, \check{\mathcal{E}}\}$, augmented features $\check{X}$

create CCnode set $\dot{\mathcal{V}} = \{\dot{v}_1, \dot{v}_2, \dots \dot{v}_k\}$ ;
augment node set $\check{\mathcal{V}} = \mathcal{V} \bigcup \dot{\mathcal{V}}$ ;
augment edge set $\check{\mathcal{E}} = \mathcal{E} \bigcup \dot{\mathcal{E}}$ ;
`// initialize cluster nodes' features`
**if** *training* **then**
  **for** $i \leftarrow 1$ **to** $k$ **do**
    $\dot{x}_i \leftarrow \frac{1}{|\mathcal{P}_i|} \sum_{v \in \mathcal{P}_i} x_v$ ;
  **end**
**end**
augment node feature set $\check{X} = X \bigcup \{\dot{x}_i\}_{i=1}^{k}$ ;
Output $\check{\mathcal{G}} = \{\check{\mathcal{V}}, \check{\mathcal{E}}, \check{X}$ ;

图聚类操作只是将节点归类，而图增广操作 $\mathbb{A}$（见算法3）则将这种归类显示地表达在图中。图增广分为两步：1. 向原图中添加簇中心节点和相应的边 2. 初始化簇中心节点特征。令簇中心节点集为 $\dot{\mathcal{V}} = \{\dot{v}_1, \dot{v}_2, \dots \dot{v}_k\}$，原节点集为 $\mathcal{V}$，那么增广之后的节点集 $\check{\mathcal{V}}$ 为

$$\check{\mathcal{V}} = \dot{\mathcal{V}} \cup \mathcal{V}$$

而与簇中心节点相连的边集 $\dot{\mathcal{E}}$ 为

$$\dot{\mathcal{E}} = \bigcup_{i=1}^{k} \{(u, v^i), (v^i, u) | u \in \mathcal{P}_i\}$$

令原边集为 $\mathcal{E}$，则增广之后的边集 $\check{\mathcal{E}}$ 为

$$\check{\mathcal{E}} = \mathcal{E} \cup \dot{\mathcal{E}}$$

将簇中心节点和相关的边加入原图可以显式地提供节点之间的关系，使得节点编码时每个节点可以利用簇中心节点和与之相连的边来获取同一簇中其他节点的信息，能够扩大学习的感受野，簇的特性也保证了只有相似的节点能够相互学习，本文利用这种选择性学习来提高节点特征的表达效果。

由于在表征增强模块中簇中心节点要与原节点一同编码，所以要进行簇中心节点特征的初始化，此过程在训练和测试部分略有不同。训练阶段中，簇中心节点的初始化方式是取簇中节点特征的均值，令原节点特征为 $X$，第 i 个簇中节点的数量为

$|\mathcal{P}_i|$，则簇中心节点 $\dot{v}_i$ 的初始特征 $\dot{x}_i$ 为

$$\dot{x}_i \leftarrow \frac{1}{|\mathcal{P}_i|}\sum_{v\in\mathcal{P}_i} x_v$$

簇中心节点的初始化特征将会被注册为表征增强模块 $\mathbb{F}$ 中的可学习参数随着该模块一起学习，学习结束后表征增强模块中包含簇中心节点特征，所以在测试阶段，簇中心节点特征的初始化不用显式地进行，只需要加载表征增强模块即可。

**Algorithm 4:** Pre-processing Module $\mathbb{C}$

**Data:** graph $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$, node features $X$
**Result:** augmented graph $\check{\mathcal{G}}=\{\check{\mathcal{V}},\check{\mathcal{E}}\}$, augmented features $\check{X}$, clusters $\{\mathcal{P}_i\}_{i=1}^{k}$

clustering using Metis $\{\mathcal{P}_i\}_{i=1}^{k} \leftarrow \mathbb{P}(\mathcal{G})$ ;
augments graph $\check{\mathcal{G}}=\{\check{\mathcal{V}},\check{\mathcal{E}}\} \leftarrow \mathbb{A}(\mathcal{G},\{\mathcal{P}_i\}_{i=1}^{k})$ ;
Output $\check{\mathcal{G}},\check{X},\{\mathcal{P}_i\}_{i=1}^{k})$ ;

总的来说，预处理模块 $\mathbb{C}$（算法4）基于图分割结果增广了原图，为后续表征增强模块提供信息更加丰富的图结构，确保节点能学习到簇的信息。

## 4.3 表征增强模块

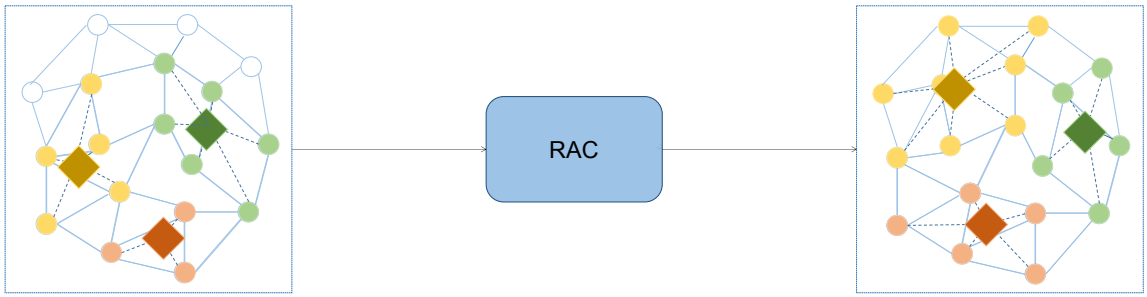


**图 2 表征增强模块的推理流程**

表征增强模块 $\mathbb{F}$ 包括可以学习的表征增强聚类器 RAC$\mathbb{M}$ 以及不需学习的簇更新模块 $\mathbb{U}$。训练阶段需要使用训练策略 $\Gamma$ 对模块 $\mathbb{F}$ 进行迭代学习，而测试推断阶段只需要运行一遍即可，方便起见本节只考虑推断阶段（如图2）中该模块运行的情况。参考预处理模块 $\mathbb{C}$ 的数学表示，表征增强模块 $\mathbb{F}$ 可表示为

**Algorithm 5:** Feature Augmentation Module $\mathbb{F}$

**Data:** augmented graph $\check{\mathcal{G}}$ augmented features $\check{X}$
**Result:** updated augmented graph $\tilde{\mathcal{G}}$, node representation $\tilde{X}$, prediction $\tilde{Y}$ ,updated clusters $\{\tilde{\mathcal{P}}_i\}_{i=1}^{k}$

$\tilde{X}, \tilde{Y}, \{\tilde{\mathcal{P}}_i\}_{i=1}^{k} \leftarrow \mathbb{M}(\check{\mathcal{G}}, \check{X})$ ;
$\tilde{\mathcal{G}} \leftarrow \mathbb{U}(\{\tilde{\mathcal{P}}_i\}_{i=1}^{k})$ ;
Output $\tilde{\mathcal{G}}, \check{X}, \tilde{Y}, \{\tilde{\mathcal{P}}_i\}_{i=1}^{k}$ ;

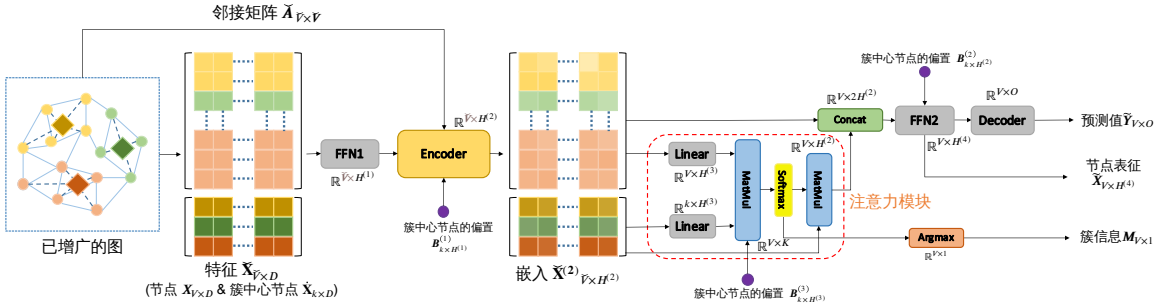


**图 3 表征增强聚类器（RAC）结构**

### 4.3.1 表征增强聚类器 RAC

表征增强聚类器 $\mathbb{M}$ 输入增广图，输出预测值、增强的节点表征和更新的簇，在训练阶段，簇中心节点的初始特征 $\{\dot{x}_i\}_{i=1}^{k}$ 被注册到模型中学习，使之能学习到全图的簇信息，之后用该簇信息进行推理。

如图3（放大版见附录中图19）所示，RAC 模型主要由两个向前传播模块（FFN）、编码器（Encoder）、注意力模块（attention module）和解码器（Decoder）构成，整个模型架构可以由它们分割成五层，详细的数学表示见算法6。在图中，特征矩阵 $\check{X}_{\check{V}\times D}$ 中每一行代表一个节点的特征向量，每个特征向量的颜色代表所属簇，最后 $k$ 个向量（图中 $k=3$）为簇中心节点的特征，之后的嵌入矩阵也是同样的含义。图中每个模块都附有该模块输出空间 $\mathbb{R}$，如果处于第 $l$ 层，那么该层的输出的维度为 $H^{(l)}$。

在不同层之间还会为簇中心节点的嵌入加上偏置以提高簇中心节点的学习效果。令第 $l$ 层的张量为 $\check{X}^{(l)} \in \mathbb{R}^{\check{V}\times H^{(l)}}$，其中 $H^{(l)}$ 表示该层的嵌入的维度，那么向前传播模块可以写为

$$FFN(\check{X}^{(l+1)}) = ELU(LN(Linear(\check{X}^{(l)})))$$

其中 $Linear(\cdot)$ 为全连接层用于空间维度的转换；$LN(\cdot)$ 为层标准化用于提高模型的泛化能力与鲁棒性，同时由于层标准化与样本数量无关所以非常适合节点（样本）数量不定的图结构；$ELU(\cdot)$ 为 ELU 激活函数。本表征增强模型是插件式模型，其中的编码器可以是其他现有模型，只需要将现有模型的输入和输出维度微调即可插入本模型进行运行。由于本模型利用簇信息学习，学习的过程与图的划分等因素相关，所

以编码器学习到的参数与其原本学习到的不同。针对不同任务解码器会有所不同，本文使用全连接层进行输出。

对于直推式学习，模型训练过程中可见测试数据，所以在训练时便可提前将测试节点归于不同簇中，然而作为归纳式模型，训练时只可为训练数据划分簇，所以如何将新加入的测试数据划分到已知的簇中是一个问题。针对这个问题，参考 Transformer[15]和 GAT[9]等工作，本文设计了注意力模块。注意力模块会计算每个节点与各个簇中心节点的注意力。令节点嵌入为 $\mathbf{H} \in \mathbb{R}^{V \times H}$，簇中心节点嵌入为 $\dot{\mathbf{H}} \in \mathbb{R}^{k \times H}$，其中 $H$ 为嵌入维度，参考 Transformer[15]，设计两个可学习矩阵 $\mathbf{W}, \dot{\mathbf{W}} \in \mathbb{R}^{H \times H}$ 用于投影嵌入，则注意力矩阵 $A \in \mathbb{R}^{V \times k}$ 的计算如下：

$$\mathbf{E} = \mathbf{H}\mathbf{W},\ \dot{\mathbf{E}} = \dot{\mathbf{H}}\dot{\mathbf{W}}$$
$$A = softmax(\mathbf{E}\dot{\mathbf{E}}^T)$$

得到的注意力有两个作用。首先用于更新簇，每个节点寻找最大注意力的簇中心节点，然后被重新划分于对应簇中，得到更新的聚类结果 $\{\mathcal{P}_i\}_{i=1}^k$：

$$\{\mathcal{P}_i\}_{i=1}^k = Argmax(A)$$

其二是进行节点表征增强，先根据注意力矩阵计算簇中心节点嵌入的加权平均矩阵 $\dot{\mathbf{H}}_{avg} \in \mathbb{R}^{V \times H}$，将该矩阵与节点嵌入进行拼接后输入向前传播模块 $FFN_2$ 获得增强的节点特征 $\mathbf{H}_{aug} \in \mathbb{R}^{V \times 2H}$：

$$\dot{\mathbf{H}}_{avg} = A\dot{\mathbf{H}}$$
$$\mathbf{H}_{aug} = FNN_2(concat(\mathbf{H}, \dot{\mathbf{H}}_{avg}))$$

$concat(\cdot,\cdot)$ 在特征维度拼接矩阵，因 $\mathbf{H}$ 大小为 $V \times H$，$\dot{\mathbf{H}}_{avg}$ 大小为 $V \times H$，所以拼接后的维度为 $V \times 2H$。

### 4.3.2 簇更新

簇的更新展现在图上即是节点与簇中心节点的连边发生改变。在获得更新的簇 $\{\widetilde{\mathcal{P}}_i\}_{i=1}^k$ 之后簇更新 $\mathbb{U}$ 即为边的更新：

$$\dot{\mathcal{E}} \leftarrow \bigcup_{i=1}^{k} \{(u, v^{(i)}), (v^{(i)}, u) | u \in \widetilde{\mathcal{P}}_i\}$$
$$\check{\mathcal{E}} = \mathcal{E} \cup \dot{\mathcal{E}}$$

**Algorithm 6:** Representation Augmentation Clusterer (RAC) $\mathcal{M}$

**Data:** augmented graph $\check{\mathcal{G}}$, augmented features $\check{X}$
**Result:** node representation $\tilde{X}$, prediction $\tilde{Y}$ ,updated clusters $\{\tilde{\mathcal{P}}_i\}_{i=1}^k$

gets the first-layer hidden variables $\check{\mathbf{H}}^1 \leftarrow FFN_1(\check{X})$ ;
encodes hidden variables $\check{\mathbf{H}}^2 \leftarrow Encoder(\check{\mathbf{H}}^1)$ ;
splits nodes' embedding and cluster nodes' embedding $\mathbf{H}^2, \dot{\mathbf{H}}^2 \leftarrow \check{\mathbf{H}}^2$ ;
gets attention $A$ from attention module ;
gets weighted cluster node embedding $\dot{\mathbf{H}}_{avg} \leftarrow A\dot{\mathbf{H}}$ ;
concats embedding in feature dimension $\check{\mathbf{H}}^3 \leftarrow concat(\mathbf{H}, \dot{\mathbf{H}}_{avg})$ ;
gets node representation $\tilde{X} \leftarrow FFN_2(\check{\mathbf{H}}^3)$ ;
gets prediction $\tilde{Y} \leftarrow Decoder(\tilde{X})$ ;
gets updated clusters $\{\tilde{\mathcal{P}}_i\}_{i=1}^k \leftarrow argmax(A)$ ;
Output $\tilde{X}, \tilde{Y}, \{\tilde{\mathcal{P}}_i\}_{i=1}^k$ ;

## 4.4 模型解释与训练

在大规模图训练时小批量学习很常见，现有的模型通常在原图中通过多次采样或者图分割的方式获取子图进行训练，这种情况下，无论采样或分割策略如何设计，每一批子图中都只包含该批的信息，其获取的总是局部信息，其用小图的数据分布近似原图的数据分布，这种方式会带来无法忽视的偏差。为此，本算法设计了全局的簇中心节点，簇中心节点对应全图图分割结果并接入原图，此时簇中心节点编码后能够抽象与概括各个簇的信息，簇中心节点集实际上即可看作全图的特征。为了缓解传统小批量学习带来的偏差，簇中心节点将注册到表征增强聚类器中，原节点在进行编码时便可以利用对应簇中心节点的信息，而簇中心节点包含全局信息，所以即使在小批量中，每个节点仍然具备全局视野并利用相关信息，从而缓解了小批量学习中数据分布的偏差。

为了设计具体的神经网络结构，本文参考了传统聚类步骤，将表征增强聚类器设计为一个类聚类器。以经典的 K-Means 算法为例，其聚类的过程为：1. 设置聚类个数 $k$，初始化质心位置 2. 计算各个样本与质心的距离 3. 每个样本归入最近的质心对应的簇，簇由此更新 4. 使用新簇中的样本计算每个簇的质心 5. 重复步骤 2 到 4，直到质心不再变动或者达到最大迭代数（如图4）。


**图 4 一般聚类流程**

不同于 K-Means 只有一个空间，本模型 $\mathbb{M}$ 含有两个空间：原特征空间、嵌入空间。以图3中所示，节点特征在未经过 $FFN_1$ 层之前处于原特征空间，在经过 $FFN_1, Encoder$ 的编码之后进入嵌入空间，在嵌入空间中通过注意力模块使用内积衡量质心和样本

的相似度，越相似内积越大，从而距离越近。由于这两个空间的存在，质心也分为特征空间中的和嵌入空间中的：K-Means 中的质心对应嵌入空间质心，该质心的更新在距离计算之前，随着编码隐式地进行；特征空间质心即为簇中心节点，其位置随着模型学习，嵌入空间质心的生成依赖于簇中心节点的位置以及簇中心节点与节点的连接关系。总的类比来看，节点为样本、簇中心节点为特征空间质心、编码器进行嵌入空间质心的更新、注意力模块用于计算样本与嵌入空间质心的距离、$Argmax$ 操作和后续的簇更新模块 $\mathbb{U}$ 用于更新簇。训练阶段本模型会不断迭代学习特征空间质心的位置、如何从特征空间转换到嵌入空间以及如何计算距离，推理阶段不需迭代可以一步获取结果，聚类流程如图5。基于一般的聚类流程设计，本模型有一定的解释性。需要注意的是，本模型最终目的还是针对特定节点任务而非生成最好的聚类结果，此聚类结果只是模型的副产品，以上只是从聚类的角度对模型进行分析。

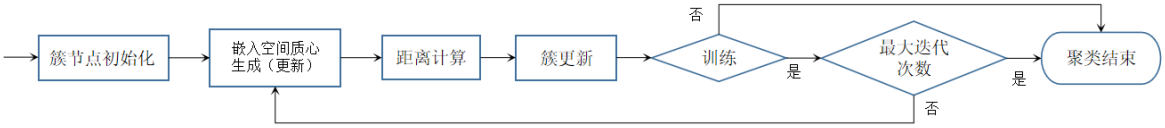


**图 5 本模型聚类流程**

实际训练过程中如果注意力模块和编码器一同学习，由于编码得到的嵌入也不稳定，容易导致大部分节点被归于同一簇。考虑到注意力模块可学习参数一般较编码器少的多，注意力模块的训练策略有所不同，其不在每批进行更新，而是隔 $d$ 批更新。令表征增强聚类器 RAC 的可学习参数为 $\{\mathcal{W}_i\}_{i=1}^{5}$，其中 $FFN_1, Encoder, FNN_2, Decoder$ 的可学习参数为 $\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_4, \mathcal{W}_5$，注意力模块的可学习参数为 $\mathcal{W}_3$，那么本模型的学习策略为算法7。在实际训练中，考虑到作为编码器的原模型也需要时间学习，其在训练初期编码的特征可能不太有效，这时如果注意力模块一同学习可能会导致双方的学习效率低甚至学习到错误信息而过早进入局部最优，因为相对于编码器，注意力模块通常含有较少的学习参数，所以提出了针对注意力模块的热身策略，即先锁定注意力机制进行训练，在 $w$ 个迭代轮次之后将其解锁使之随着整个模型一同训练，$w$ 需要人工设定（算法7中未体现热身策略）。

## 4.5 复杂度分析

本节将依次从聚类操作、图增广操作以及表征增强模块分析本算法空间和时间上的复杂度。聚类操作使用的是 Metis 算法，Metis 算法先随机获得一组节点的匹配，使得每个节点最多与另一个节点匹配上，匹配结果是边集，匹配过程对边进行操作，所以该过程的时间和空间复杂度为 $O(|E|)$。配到一起的节点看作为一个节点，以此来粗化原图，目的是减小图的规模，多次匹配和粗化后获得一个小图，另进行了 $b$ 次迭代，那么此过程的时间复杂度为 $O(b|E|)$。接下来进行分割操作，交换节点使得匹配变化，使用一个启发式的指标评价交换后的优劣，等达到最大交换次数或者评价指标变化幅度小于阈值则停止交换。然后进行细化即粗化的逆过程，将合并的节点拆开，然后进行改善操作，思想与分割操作基本一致，也是交换节点贪心地进行改善，

**Algorithm 7:** Training Strategy $\Gamma$

**Data:** augmented graph $\check{\mathcal{G}}$, augmented features $\check{X}$, label $Y$
**Result:** graph with trained clusters $\tilde{\mathcal{G}}$, model's parameters $\{\mathcal{W}_i\}_{i=1}^5$

$d \leftarrow$ updating gap of attention module;
$g_{attn} \leftarrow 0$ ;
**for** $i \leftarrow 1$ **to** $max\_iter$ **do**
    runs RAC $\tilde{X}, \tilde{Y}, \{\tilde{\mathcal{P}}_i\}_{i=1}^k \leftarrow \mathbb{M}(\check{\mathcal{G}}, \check{X})$ ;
    updates clusters in the graph $\tilde{\mathcal{G}} \leftarrow \mathbb{U}(\{\tilde{\mathcal{P}}_i\}_{i=1}^k)$ ;
    calculates gradients $g, g'_{attn} \leftarrow \nabla\mathbb{L}(Y, \tilde{Y})$ ;
    updates $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_4, \mathcal{W}_5\}$ using gradient $g$ ;
    $g_{attn} \leftarrow g_{attn} + \frac{1}{d} g'_{attn}$ ;
    **if** $i\%d = 0$ **then**
        updates $\mathcal{W}_3$ using gradient $g_{attn}$ ;
        $g_{attn} \leftarrow 0$ ;
    **end**
**end**
Output $\tilde{\mathcal{G}}, \{\mathcal{W}_i\}_{i=1}^5$ ;

以上步骤一直重复直到图恢复到原图。以上操作有不同变种，总的来说时间和空间复杂度为 $O(|E|)$。图增广操作将簇中心节点和相应的边加入原图中，由于每个节点都必须且只能属于一个簇，所以该操作的时间和空间复杂度为 $O(|V|)$。聚类和图增广作为预处理都在 CPU 中进行，所以实际上空间上的消耗可以忽略不计，其总的时间复杂度为 $O(|V|+|E|)$，实际运行中由于边数量通常会比节点数量大一到两个左右数量级且 Metis 算法时间复杂度有一定大小的常数项，所以预处理的时间大部分还是在聚类，不过百万节点级别的大图仍然可以在十几秒内完成聚类，且每次训练只需进行一次预处理，所以这部分的时间消耗基本上也能忽略。

表征增强模块包括 RAC 和簇更新模块。簇更新模块更新节点与簇中心节点的连接情况，时间复杂度为 $O(|V|)$。神经网络的复杂度主要在编码器和注意力模块，编码器是任意模型所以复杂度不可控，令其为 $O(Encoder)$。注意力模块需要将节点嵌入与簇中心节点嵌入相乘，另簇中心节点数量为 $q$，则该模块时间和空间复杂度为 $O(q|V|)$。那么表征增强聚类器的时间与空间复杂度为 $max(O(Encoder),O(q|V|))$，这也是表征增强模块总的复杂度。一般来说，编码器的复杂度不会低于 $O(|V|)$，所以该模块的复杂度可以写为 $O(Encoder)$, 其还是基本由插入的外部模型决定。

因此，基于理论和实际情况，本算法带来的格外时间和空间复杂度不会对加入的模型造成明显的影响。

## 5. 实验

本文着重研究节点级任务中的节点分类任务，使用 OGB[29] (Open Graph Benchmark) 中的 ogbn-arxiv 和 ogbn-proteins 数据集（见表1），它们拥有超过十万个节点和百万条边的图，适合用作大规模图进行模型训练与测试。数据集 ogbn-arxiv 为论文之

间的引用关系图，每个节点代表“论文”对象，节点之间的边代表“论文”之间的“引用”关系，任务是利用“引用”关系和“论文”节点特征判断“论文”所属的学科，一共 40 种学科；数据集 ogbn-proteins 为蛋白质关系图，每个节点代表“蛋白质”对象，边代表蛋白质之间的化学联系，任务是判断“蛋白质”是否拥有指定的 112 个功能。以上数据集均为同构图即每个节点都代表同种对象、每个边代表同种关系，同时本文将它们都看作无向图进行处理。

因为本算法作为插件用于增强模型效果，所以使用比较简单和经典图神经网络进行对比，包括 GCN、GraphSAGE 和 Nodeformer，并使用全连接模型作为基线。本文实验平台为 11GB 显存的 NVIDIA GeForce RTX 2080 Ti 显卡以及 2.60GHz 频率的 Intel(R) Xeon(R) Gold 6240 处理器。

**表 1 数据集**

| 数据集 | 节点数 | 边数 | 任务类型 | 指标 |
|---|---|---|---|---|
| ogbn-arxiv | 169,343 | 1,166,243 | 多任务分类 | accuracy |
| ogbn-proteins | 132,534 | 39,561,252 | 多任务二分类 | ROC-AUC |

## 5.1 实验效果对比

本节先获得待测试模型在 ogbn-arxiv 和 ogbn-proteins 数据集上的测试结果作为对照组，然后分别给它们添加 ClusterALL 算法再次测试获得实验数据进行对比，表2和表3分别展示了两个数据集的实验结果，其中“MLP + ClusterALL, k=3”表示 MLP 模型结合 ClusterALL 且聚类数量为 3。MLP 作为基线，不考虑边只对节点进行编码，Nodeformer 模型使用线性注意力机制可以编码全图的节点特征以及图结构，GCN 模型是经典的图神经网络一定程度上能反映大部分模型的机制，GraphSAGE 基于采样进行编码，其中除了 GCN 是直推式模型外，其他都是归纳式模型。从表2和3可见，在添加了 ClusterALL 之后，所有模型的效果都得到了提升，可见本算法可以很好地与直推式和归纳式模型进行结合，能够有针对性地增强模型所编码得到的节点特征从而提高其在不同任务中的效果。

**表 2 ogbn-arxiv 测试结果**

| Model | Accuracy (%) |
|---|---|
| MLP | 56.12 |
| MLP + ClusterALL, k=3 | **57.71** |
| Nodeformer | 67.92 |
| Nodeformer + ClusterALL, k=5 | **68.66** |
| GCN | 72.10 |
| GCN + ClusterALL, k=3 | **72.85** |
| GraphSAGE | 71.87 |
| GraphSAGE + ClusterALL, k=5 | **72.64** |

**表 3 ogbn-proteins 测试结果**

| Model | ROC-AUC (%) |
|---|---|
| MLP | 72.08 |
| MLP + ClusterALL, k=10 | **73.58** |
| Nodeformer | 76.84 |
| Noderformer + ClusterALL, k=10 | **77.78** |
| GCN | 72.49 |
| GCN + ClusterALL, k=3 | **73.28** |
| GraphSAGE | 77.36 |
| GraphSAGE + ClusterALL, k=2 | **78.37** |

## 5.2 消融实验

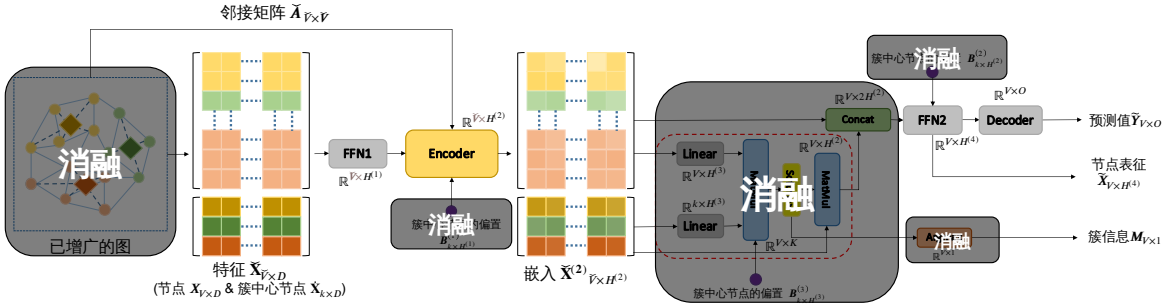


**图 6 模型消融示意图**

本算法的核心模块在于预处理以及 RAC 中的注意力模块，本节通过去除核心模块进行消融实验来验证本算法的有效性。消融部分具体见图6：RAC 模型输入的图为原图，不进行预处理操作即图聚类和簇中心节点的添加；注意力模块被移除，编码得到的节点表征直接进行解码；由于不存在簇中心节点，所以与簇中心节点相关的可学习偏置被移除。由于以上三个部分环环相扣必须进行预处理后才有簇中心节点以及之后的编码与簇更新，所以只能三个部分同时消融不能进一步分解消融。消融实验使用 MLP、Nodeformer、GCN 和 GraphSAGE 模型在 ogbn-arxiv 数据集中进行，评价指标为准确度（%），实验结果见表4。表的中间列“with ClusterALL, k=0”代表消融的 ClusterALL 算法即图6所展示，进行对比的非消融模型使用的聚类数量 $k$ 统一为3，此时的聚类数量不一定为最优的。即使非最优聚类参数，ClusterALL 的加入也能一定程度提高原模型的效果，说明本文的基于深度聚类的核心思想有效，簇中心节点带来的全局的、跨批的信息有助于节点的表征。

## 5.3 超参数测试

为进一步探究超参数聚类数量 $k$ 对本算法 ClusterALL 效果的影响，本节在 ogbn-arxiv 数据集中使用不同 $k$ 测试各个模型，得到如表11所示的结果，表中虚线为单独使用原模型的效果，实线为添加 ClusterALL 获得的结果。总体来看，在聚类数量 $k$

表 4 消融实验测试结果

| Model | with ClusterALL, k=0 | with ClusterALL, k=3 |
|---|---|---|
| MLP | 56.92% | **57.71%** |
| Nodeformer | 67.83% | **67.94%** |
| GCN | 72.72% | **72.85%** |
| GraphSAGE | 72.38% | **72.40%** |

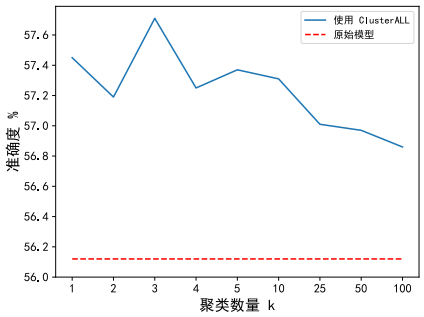


图 7 MLP 模型

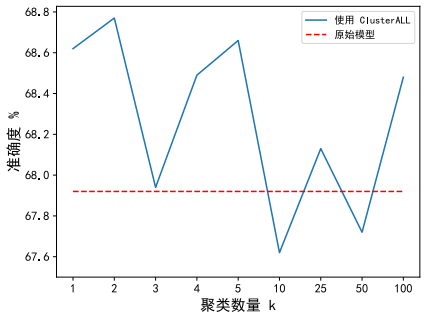


图 8 Nodeformer 模型

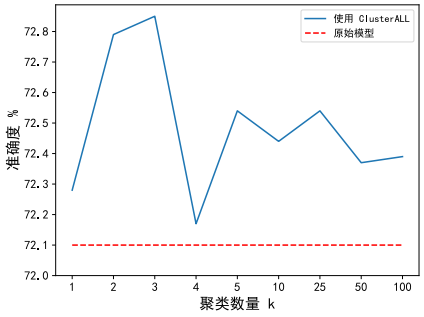


图 9 GCN 模型

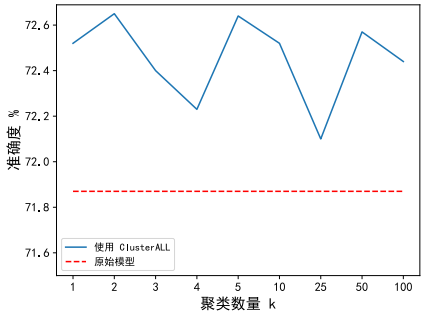


图 10 GraphSAGE 模型

图 11 聚类数量在不同模型中对准确度影响

在 100 以内时，大部分情况下添加 ClusterALL 的模型都能获得更好的效果，随着聚类数量从小到大变化，ClusterALL 的效果先增加后减少。对于大多数模型，聚类数量不用很大（通常在 10 以内）便能获得几乎最好的结果，为此本文的解释是聚类数量过大时可学习参数较多，聚类情况较复杂，模型无法在有效时间内完全较好的编码与聚类学习，从而导致效果下降；当聚类数量较小时，模型可以更专注于编码和特定几个聚类特征的学习，此时每个簇的规模很大，簇学习到的是更加抽象的特征，而簇中心节点的存在使得同簇内的节点可以相互学习，所以节点能够在图中有选择地获取全局信息，这样有利于节点特征编码。Nodeformer 模型在部分聚类数量下获得了较差的结果，这可能因为该模型本身具备全局感受野，而 ClusterALL 的簇中心节点也能从全图的特定节点即簇中抽象特征，当聚类节点不合适时，高度抽象的簇特征可能会扰乱原模型的全局编码效果。

如小节4.4提到的，本算法还包含用于训练注意力模块的超参数更新间隔 $d$。本节使用 GraphSAGE 模型在 ogbn-arxiv 数据集中测试更新间隔对准确度的影响，实验结果如图12所示。图中虚线表示不更新注意力模块，或者可以看作更新间隔为无穷大 $d=\infty$，实线为更新间隔改变后准确率的变化曲线。从图中可见准确率随着更新间隔增大先提高后减小，当更新间隔在 200 左右时准确率能达到最大，说明本文提出的训练策略对于注意力模块的学习有效，验证了小节4.4中的猜想。

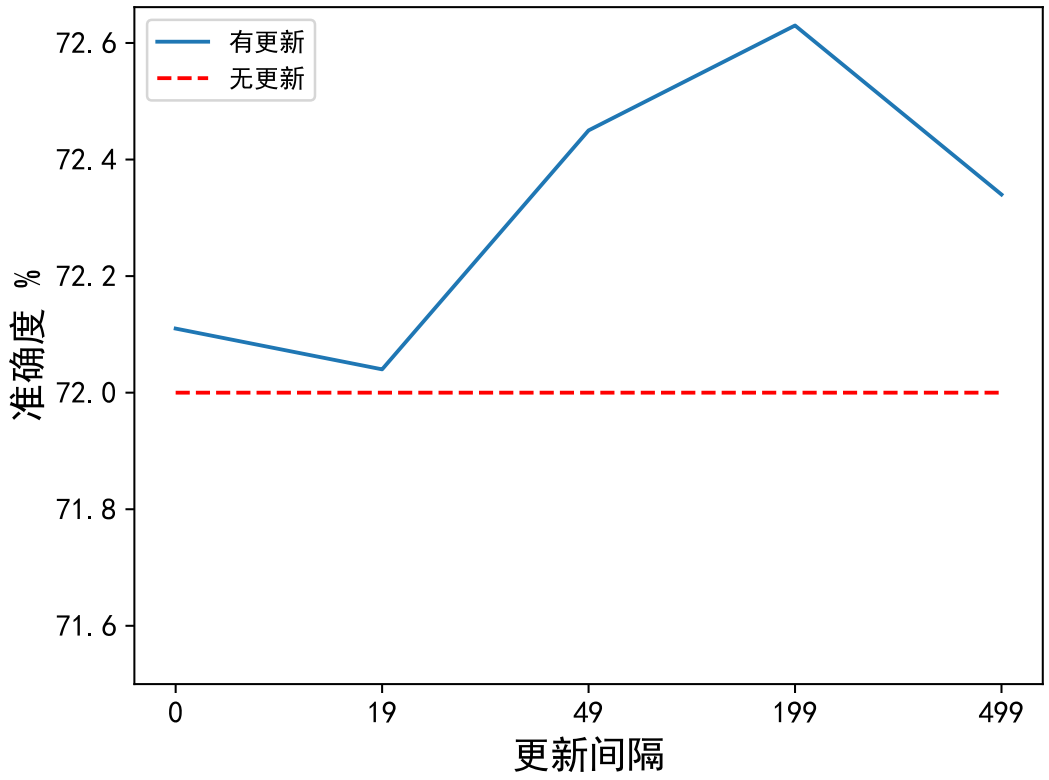


**图 12 注意力模块更新间隔对准确度的影响**

## 5.4 聚类效果分析

考虑到以节点任务为导向的情况下，聚类结果通常没有外部标签进行监督，所以本文使用内部指标进行聚类效果的评估。内部评价指标通过计算聚类结果中同类中样本紧密程度以及不同类间样本的疏离程度评估聚类的优劣，常用的指标有轮廓系数[30]、CH 值[31]和戴维森宝丁指数[32]（DBI），由于轮廓指数复杂度太高不适用于大规模图所以本文使用 CH 值和 DBI 值进行评价。CH 值范围为 $[0,+\infty)$，CH 值越大表示聚类效果越好；DBI 值范围 $[0,+\infty)$，该值越小聚类效果越好。

本算法加入的簇中心节点以及可学习的注意力模块能够帮助原模型在针对特定任务的情况下获得更加便于聚类的节点表征。为验证以上说法，本节使用 GraphSAGE 模型在 ogbn-arxiv 数据集上进行实验，分别得到 ClusterALL（$k=5$）添加前后训练

时的节点表征，然后使用 sklearn 库中的小批量 KMeans 算法进行同聚类数量的聚类，最后利用 DBI 和 CH 值评估，获得图15展示的结果。首先单独看原模型的情况，随着模型训练，DBI 值先下降后持续上升，CH 值先上升后持续下降，编码得到的节点表征随着学习的进行整体来说是越来越不适合进行聚类；添加 CLusterALL 算法后，虽然不断的学习仍然导致节点特征越来越不适于聚类，但是这种趋势得到了缓解，整体的聚类效果优于原始模型。因此，本算法能使节点学习到聚类信息从而得到更加适合于聚类的表征，这种表征因为是任务导向的所以还能增强模型在相应任务中的效果（如小节5.1所展示）。

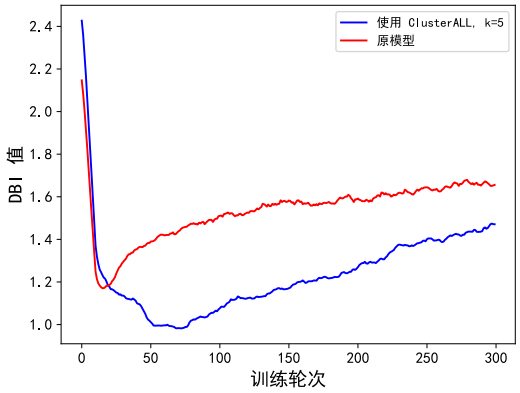


图 13 DBI 对比

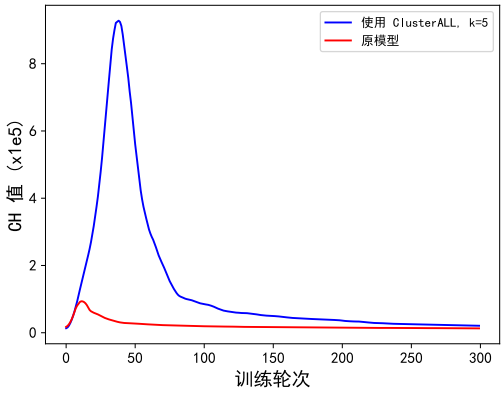


图 14 CH 值对比

图 15 ClusterALL 添加前后节点聚类效果对比

图16展示了在 GraphSAGE 中添加 ClusterALL（$k = 5$）后，在 ogbn-arxiv 数据集中的聚类效果。该聚类效果是节点特征降维为 2 维后进行聚类，同时该 2 维特征作为节点的坐标显示在图中。为了进一步探究本算法聚类如何影响节点特征的编码，图17将图16中红色簇单独提出并保留原图的边连接。可见同一簇中的节点也不是完全连通，部分节点是孤点，这说明由于簇中心节点的存在，节点可以用更大的感受野有选择地捕获相似节点的信息。

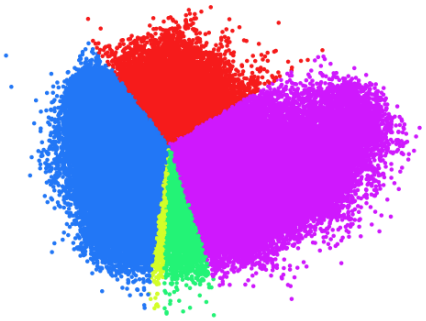

图 16 聚类数量为 5 时的聚类结果

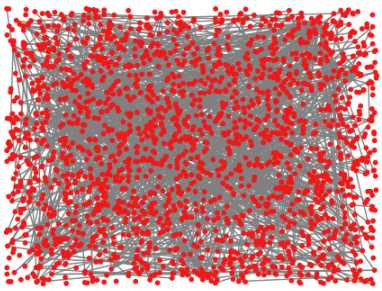

图 17 同簇中心节点的连接情况

## 6. 结论

本文提出了基于深度聚类的可以增强节点表征的插件式模型，该模型使节点真正获取全图感受野并有选择地聚合图中有效的信息，同时该模型具备可扩展性能适用于大规模图。本算法的核心步骤为聚类和加入可学习的簇中心节点与相应边，同时注意力模块使本模型具备归纳能力。簇中心节点连接在高度抽象的维度中相似的节点，这种情况下同簇的节点可能并不相连甚至不连通，它们可能分布在大图中任意位置，使得各个节点能够感知到更广的信息。更重要的是，簇中心节点储存于特征增强模型中随模型学习，位于图中各个位置的同簇中心节点将不断训练相应簇中心节点，从而使簇中心节点能够使用全局信息表征该簇，同时具有簇中心节点又能提供各个节点包含全局感受野的簇特征，使得即使进行小批量学习，各个节点仍然可以从簇中心节点获取来自全图的特定信息。此外，为了使模型更具可解释性，本文从一般的聚类步骤设计与解释神经网络结构。最后，本模型在多个数据集和模型上进行测试，进行消融实验并探究了各个超参数对模型的影响情况，验证了本模型的有效性与可扩展性；通过聚类的可视化对本模型的假设与猜想进行了验证。

# 参考文献


[1] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[2] PEROZZI B, AL-RFOU R, SKIENA S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 701-710.

[3] TANG J, QU M, WANG M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th international conference on world wide web. 2015: 1067-1077.

[4] GROVER A, LESKOVEC J. node2vec: Scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

[5] WANG D, CUI P, ZHU W. Structural deep network embedding[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 1225-1234.

[6] RIBEIRO L F, SAVERESE P H, FIGUEIREDO D R. struc2vec: Learning node representations from structural identity[C]//Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. 2017: 385-394.

[7] KONG K, LI G, DING M, et al. Flag: Adversarial data augmentation for graph neural networks[J]. arXiv preprint arXiv:2010.09891, 2020.

[8] HAMILTON W L, YING R, LESKOVEC J. Inductive Representation Learning on Large Graphs[Z]. 2018.

[9] VELIČKOVIĆ P, CUCURULL G, CASANOVA A, et al. Graph Attention Networks [Z]. 2018.

[10] BELKIN M, NIYOGI P. Laplacian eigenmaps for dimensionality reduction and data representation[J]. Neural computation, 2003, 15(6): 1373-1396.

[11] DWIVEDI V P, LUU A T, LAURENT T, et al. Graph neural networks with learnable structural and positional representations[J]. arXiv preprint arXiv:2110.07875, 2021.

[12] WANG H, YIN H, ZHANG M, et al. Equivariant and stable positional encoding for more powerful graph neural networks[J]. arXiv preprint arXiv:2203.00199, 2022.

[13] SHI Y, HUANG Z, FENG S, et al. Masked label prediction: Unified message passing model for semi-supervised classification[J]. arXiv preprint arXiv:2009.03509, 2020.

[14] WU Q, ZHAO W, LI Z, et al. NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification[J]., 2022.

[15] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[16] YING C, CAI T, LUO S, et al. Do transformers really perform badly for graph representation?[J]. Advances in Neural Information Processing Systems, 2021, 34: 28877-28888.

[17] ZHOU H, ZHANG S, PENG J, et al. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting.[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2022: 11106-11115.

[18] WU H, XU J, WANG J, et al. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting[Z]. 2022.

[19] CHOROMANSKI K, LIKHOSHERSTOV V, DOHAN D, et al. Rethinking Attention with Performers[Z]. 2022.

[20] YING R, HE R, CHEN K, et al. Graph Convolutional Neural Networks for Web-Scale Recommender Systems[J]., 2018.

[21] CHEN J, MA T, XIAO C. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling[Z]. 2018.

[22] CHIANG W L, LIU X, SI S, et al. Cluster-GCN[J]., 2019.

[23] ZENG H, ZHOU H, SRIVASTAVA A, et al. GraphSAINT: Graph Sampling Based Inductive Learning Method[J]., 2020.

[24] KARYPIS G, KUMAR V. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs[J]. SIAM Journal on Scientific Computing, 1998, 20(1): 359-392.

[25] DHILLON I S, GUAN Y, KULIS B. Weighted Graph Cuts without Eigenvectors A Multilevel Approach[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(11): 1944-1957.

[26] WANG C, PAN S, HU R, et al. Attributed graph clustering: A deep attentional embedding approach[J]. arXiv preprint arXiv:1906.06532, 2019.

[27] NAZI A, HANG W, GOLDIE A, et al. Gap: Generalizable approximate graph partitioning framework[J]. arXiv preprint arXiv:1903.00614, 2019.

[28] BLONDEL V D, GUILLAUME J L, LAMBIOTTE R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10): P10008.

[29] HU W, FEY M, ZITNIK M, et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs[J]. arXiv preprint arXiv:2005.00687, 2020.

[30] ROUSSEEUW P J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis[J]. Journal of Computational and Applied Mathematics, 1987, 20: 53-65.

[31] CALIŃSKI T, HARABASZ J. A dendrite method for cluster analysis[J]. Communications in Statistics, 1974, 3(1): 1-27.

[32] DAVIES D L, BOULDIN D W. A Cluster Separation Measure[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979, PAMI-1(2): 224-227.

# 附录

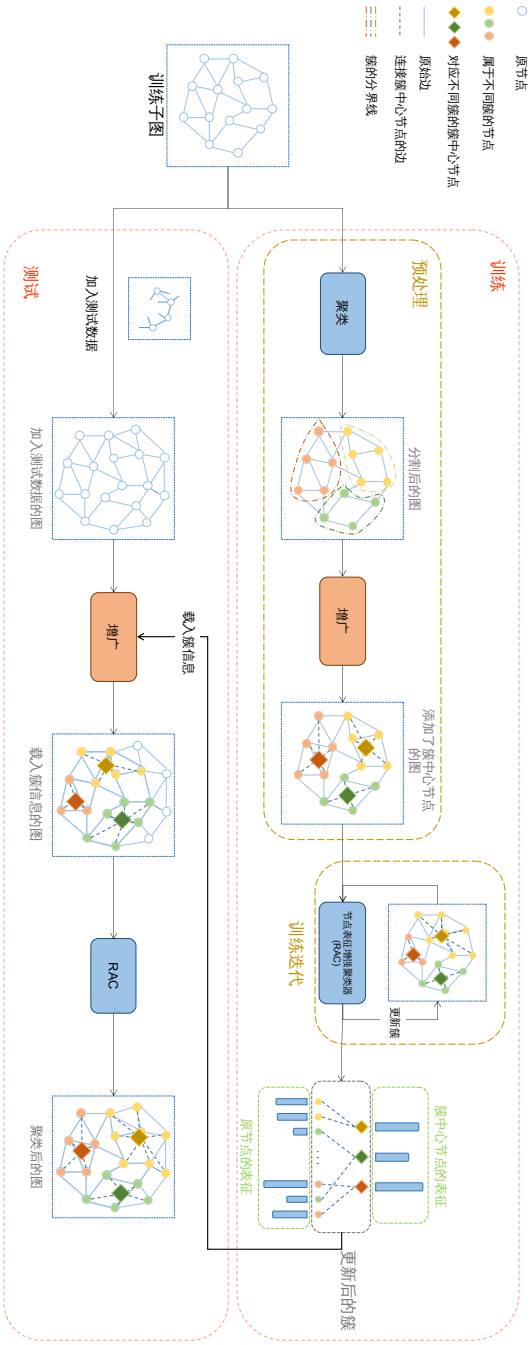


**图 18 ClusterALL 结构图放大版**

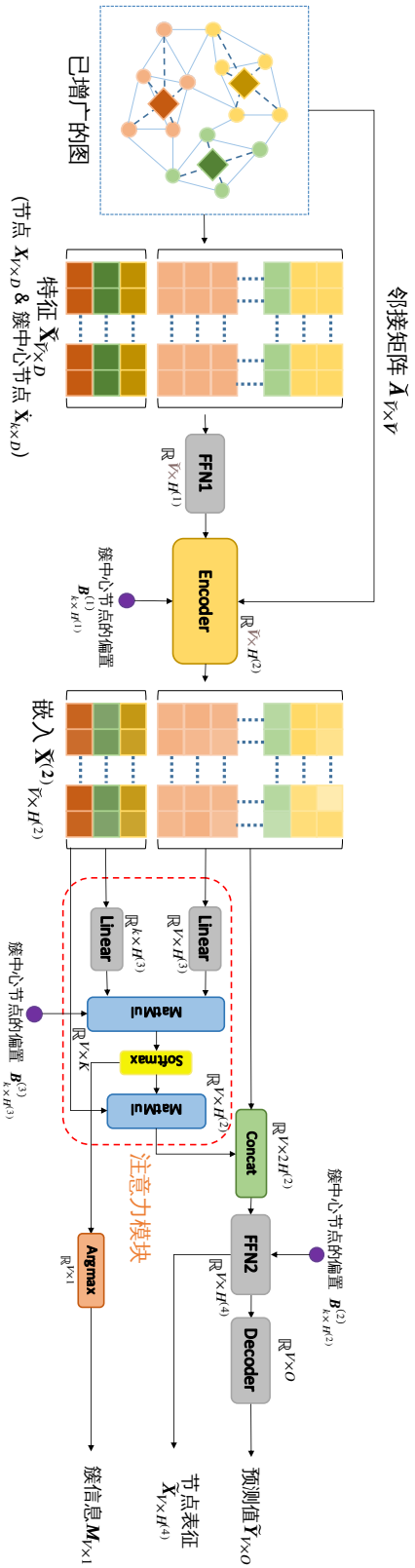


**图 19 表征增强聚类器（RAC）结构图放大版**