基于 GNN 的金融业务异常检测

赵林芯

2025

目 录

[1 选题目的 1](#_Toc204769048)

[1.1 课题动机 1](#_Toc204769049)

[1.2 主要工作内容 1](#_Toc204769050)

[2 实验数据 2](#_Toc204769051)

[2.1 数据集来源 2](#_Toc204769052)

[2.2 数据集详细描述 2](#_Toc204769053)

[2.3 数据预处理方法 4](#_Toc204769054)

[2.4 图数据建模 6](#_Toc204769055)

[2.4.1 同构图与异构图的区别 6](#_Toc204769056)

[2.4.2 构建异构关系图 7](#_Toc204769057)

[2.4.3 图构建流程 8](#_Toc204769058)

[2.4.4 图构建算法复杂度分析 13](#_Toc204769059)

[3 所采用的深度学习方法的原理 17](#_Toc204769060)

[3.1 函数的假设空间（网络结构） 17](#_Toc204769061)

[3.1.1 图神经网络基础架构 17](#_Toc204769062)

[3.1.2 多关系图建模 18](#_Toc204769063)

[3.1.3 关系内聚合机制 18](#_Toc204769064)

[3.1.4 关系间聚合机制 20](#_Toc204769065)

[3.2 目标函数 22](#_Toc204769066)

[3.2.1 损失函数设计 22](#_Toc204769067)

[3.2.2 正则化项 23](#_Toc204769068)

[3.3 优化方法 23](#_Toc204769069)

[4 实验设置、实验结果及分析 25](#_Toc204769070)

[4.1 实验环境配置 25](#_Toc204769071)

[4.2 模型参数设置 25](#_Toc204769072)

[4.3 代码运行情况 26](#_Toc204769073)

[4.3.1 训练阶段 26](#_Toc204769074)

[4.3.2 测试阶段 28](#_Toc204769075)

[4.4 模型学习过程可视化 29](#_Toc204769076)

[4.4.1 训练曲线分析 29](#_Toc204769077)

[4.4.2 特征分布可视化 30](#_Toc204769078)

[4.5 模型测试结果 31](#_Toc204769079)

[4.5.1 定量评估结果 31](#_Toc204769080)

[4.5.2 混淆矩阵分析 31](#_Toc204769081)

[4.5.3 ROC曲线分析 32](#_Toc204769082)

[4.6 嵌入向量可视化分析 33](#_Toc204769083)

[4.6.1 t-SNE降维可视化 33](#_Toc204769084)

[4.6.2 PCA降维可视化 34](#_Toc204769085)

[4.7 实验结果分析 35](#_Toc204769086)

[4.7.1 图神经网络方法优缺点分析 35](#_Toc204769087)

[4.7.2 改进方法研究 36](#_Toc204769088)

[4.8 代码说明 39](#_Toc204769089)

[4.8.1项目代码结构 39](#_Toc204769090)

[4.8.2项目类图 40](#_Toc204769091)

[5 参考文献 43](#_Toc204769092)

# 1 选题目的

## 1.1 课题动机

　　随着金融科技的快速发展和数字化支付的普及，金融交易数据呈现爆炸式增长。传统的基于规则和统计方法的欺诈检测技术在面对日益复杂的欺诈手段时显得力不从心。金融交易数据天然具有图结构特征，交易实体之间存在复杂的关联关系，这为图神经网络在金融异常检测领域的应用提供了理论基础和实践价值。

　　图神经网络作为深度学习领域的前沿技术，能够有效处理非欧几里得空间的图结构数据，通过消息传递机制捕捉节点间的复杂关系模式。相比传统方法，GNN能够同时考虑节点特征和图结构信息，在异常检测任务中表现出显著优势。

## 1.2 主要工作内容

　　本课程设计的主要工作内容包括：

　　（1）基于真实金融交易数据构建多关系图结构，将交易记录建模为图中的节点，交易关系建模为图中的边；

　　（2）设计并实现多关系图神经网络架构，包括关系内聚合器和关系间聚合器，有效融合不同类型的关系信息；

　　（3）针对金融数据的高度不平衡特性，设计标签感知聚合机制和焦点损失函数，提升模型在少数类样本上的检测性能；

　　（4）构建完整的端到端异常检测系统，包括数据预处理、模型训练、性能评估和结果可视化模块；

　　（5）通过多维度的实验分析和可视化展示，验证模型的有效性和实用性。

# 2 实验数据

## 2.1 数据集来源

　　本实验采用Kaggle平台公开的信用卡欺诈检测数据集（Credit Card Fraud Detection Dataset）。该数据集是金融异常检测领域的经典基准数据集，广泛应用于学术研究和工业实践。

**数据集名称**：Credit Card Fraud Detection

**数据来源**：Kaggle公共数据集

**下载链接**：<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

**数据提供方**：Université Libre de Bruxelles (ULB) Machine Learning Group

## 2.2 数据集详细描述

　　该数据集包含2013年9月欧洲持卡人的信用卡交易记录，具体特征如下：

　　1. **数据规模**：284,807条交易记录

　　2. **特征维度**：31个特征列

1. V1-V28：通过PCA主成分分析变换后的匿名特征，保护用户隐私
2. Time：交易时间戳，表示每笔交易与数据集中第一笔交易的时间差（秒）
3. Amount：交易金额
4. Class：标签列，0表示正常交易，1表示欺诈交易

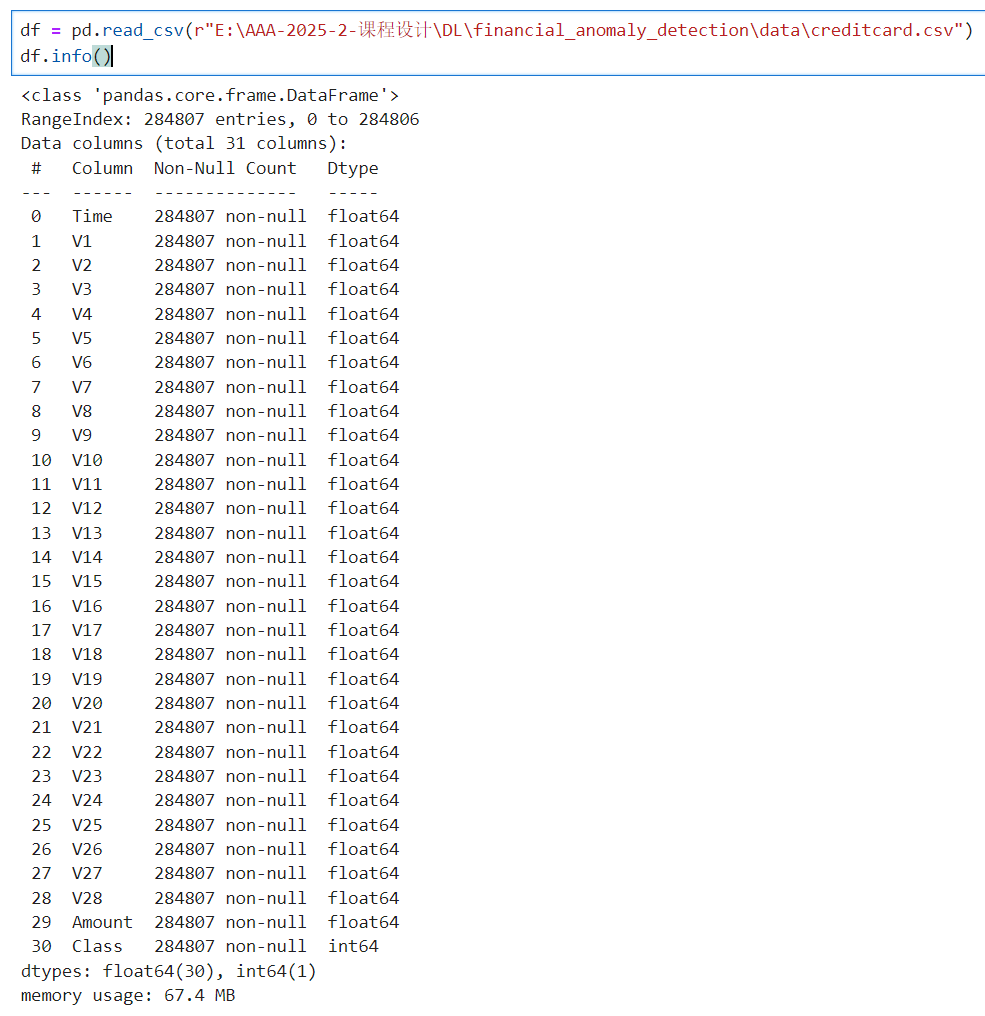


图1 Credit Card Fraud Detection数据集属性基本信息

　3. **数据质量**：

* 1. 无缺失值
  2. 数值型特征已标准化处理
  3. 数据完整性良好

　4. **标签分布**：

1. 正常交易：284,315条（99.828%）
2. 欺诈交易：492条（0.172%）
3. 数据集呈现高度不平衡特征

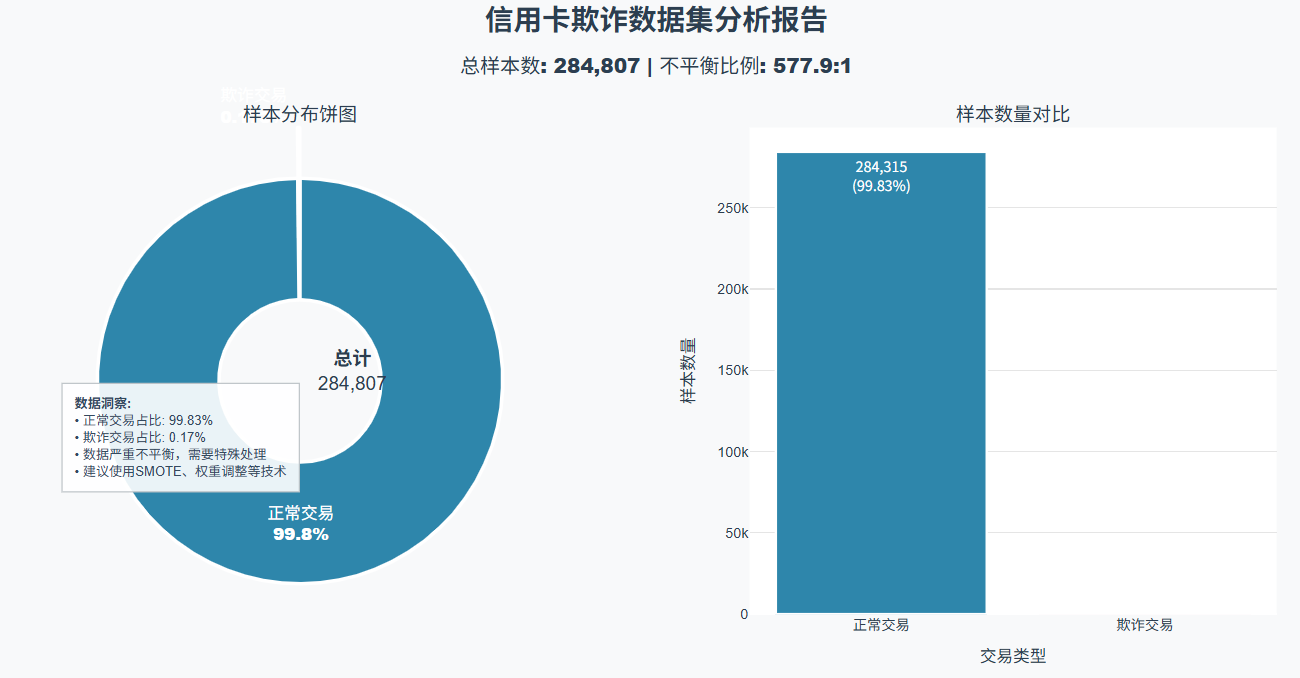


图2数据集例标签分布

## 2.3 数据预处理方法

　　为了适应图神经网络的输入要求，对原始数据进行了以下预处理：

1. **特征标准化**

对Time和Amount特征进行StandardScaler标准化，确保不同量纲特征的可比性；

|  |
| --- |
| def prepare\_features(self, data):  """准备节点特征"""  # 对于creditcard数据集，选择所有V特征和Amount、Time  if 'Class' in data.columns: # creditcard数据集  feature\_cols = [col for col in data.columns  if col not in ['Class'] and col.startswith(('V', 'Amount', 'Time'))]  target\_col = 'Class'    # ... 其他处理 ...    # 标准化特征  features = data[feature\_cols].values  features = self.scaler.fit\_transform(features) # 使用StandardScaler进行标准化    return features, data[target\_col].values |

　（2）**数据分割**：按照6:2:2的比例将数据随机分割为训练集、验证集和测试集；

在 model\_handler.py 中的分割过程：

|  |
| --- |
| # 第一次分割：训练集 vs 临时集（验证+测试）  indices = list(range(len(labels)))  idx\_train, idx\_temp, y\_train, y\_temp = train\_test\_split(  indices, labels, stratify=labels,  train\_size=self.config['train\_ratio'], # 配置文件中设置为0.6 (60%)  random\_state=self.config['seed']  )  # 第二次分割：验证集 vs 测试集  idx\_valid, idx\_test, y\_valid, y\_test = train\_test\_split(  idx\_temp, y\_temp, stratify=y\_temp,  test\_size=self.config['test\_ratio'], # 配置文件中设置为0.5 (剩余40%的一半)  random\_state=self.config['seed']  ) |

配置文件位置 ： config\financial\_credit.yml 第3-4行。

|  |
| --- |
| train\_ratio: 0.6 # 训练集占60%  test\_ratio: 0.5 # 测试集占剩余40%的一半，即20% |

　　（3）**图结构构建**：基于交易特征构建多关系图，包括金额关系、时间关系和用户行为关系。在2.4节详细展开。

## 2.4 图数据建模

### 2.4.1 同构图与异构图的区别

同构图（Homogeneous Graph） ：图中所有节点类型相同，所有边类型相同，节点和边的属性结构一致。例如社交网络中用户之间的朋友关系，所有节点都是用户，所有边都表示朋友关系。

异构图（Heterogeneous Graph） ：图中包含多种不同类型的节点或边，节点和边具有不同的属性和语义。例如学术网络中包含作者、论文、期刊等不同类型节点，以及引用、发表、合作等不同类型关系。

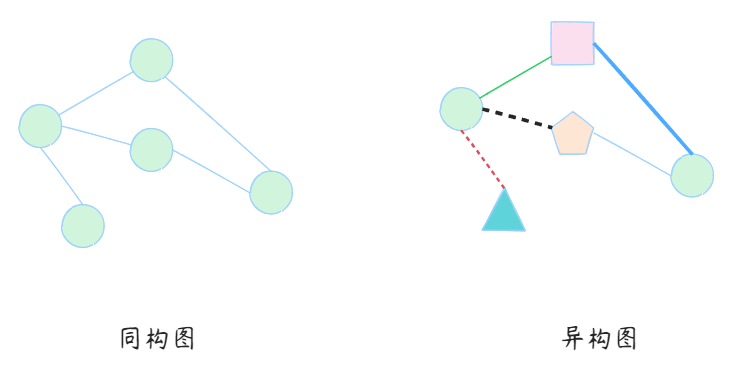


图3 同构图与异构图的对比

核心区别 ：同构图结构简单统一，适合处理单一关系类型的数据；异构图能够建模复杂的多元关系，更贴近现实世界的复杂网络结构，但计算复杂度更高。在金融异常检测等复杂场景中，异构图能够同时建模时间、金额、用户等多维关系，提供更丰富的信息表示。

### 2.4.2 构建异构关系图

基于交易特征构建多关系图，包括金额关系、时间关系和用户行为关系：

**• 金额相似性关系 ：**基于交易金额的相似度度量连接节点，通过设定阈值或使用余弦相似度等方法量化金额相似性。

**• 时间邻近性关系 ：**在预定义时间窗口内连接时间相邻的交易节点，捕获交易的时序依赖关系。

**• 用户行为相似性关系 ：**基于用户历史行为特征向量的相似度连接节点，反映用户交易模式的相似性。

以上三种关系图通过 graph\_builder.py 实现，构建了包含三种关系类型的异构图，分别是时间邻近关系、金额相似关系、用户特征关系。

1. **异构图定义：**

(1)

其中：

1. V 是节点集合（交易节点）
2. E 是边集合
3. 是关系类型集合
4. **时间关系边构建：**

(2)

其中 ( = 3600) 秒为时间窗口阈值

公式(2)定义了基于时间邻近性的边构建规则，当两个交易节点的时间差在预设窗口内时建立连接。时间窗口设置为3600秒（1小时），能够有效捕获短时间内发生的相关交易模式，这对于识别批量欺诈行为具有重要意义。

**（三）金额关系边构建：**

(3)

其中 ( = 0.1) 为相似度阈值，( = 1) 防止除零。

公式 (3) 采用相对差异度量金额相似性，通过归一化处理避免了绝对金额差异的影响。相似度阈值0.1表示金额相对差异小于10%的交易将被连接，这种设计能够识别金额相近的可疑交易模式。小常数ε的引入确保了数值计算的稳定性。

**（四）用户关系边构建：**

(4)

其中 为节点 的PCA特征向量，( = 0.8) 为相关性阈值。

公式（4）基于用户特征的皮尔逊相关系数构建边连接，PCA降维后的特征向量能够有效表征用户行为模式。相关性阈值0.8确保只有高度相似的用户行为才会建立连接，这有助于发现具有相似欺诈行为特征的用户群体。

### 2.4.3 图构建流程

**（一）图构建整体架构**

本系统采用异构图建模方法，将金融交易数据转换为图结构数据。整个图构建过程分为数据预处理、关系构建、参数优化和结构生成四个主要阶段。

**（二）数据预处理阶段**

首先，系统从creditcard.csv数据集中读取原始交易数据，该数据集包含284,807条交易记录和31个特征维度。FinancialDataProcessor类负责执行数据预处理操作，包括缺失值处理、异常值检测和数据清洗。

随后进行特征提取，选择Amount（交易金额）、Time（时间戳）以及V1-V28（PCA降维特征）作为核心特征。为确保特征的可比性，采用StandardScaler对数值特征进行标准化处理，并按照60%训练集、20%验证集、20%测试集的比例进行数据分割。

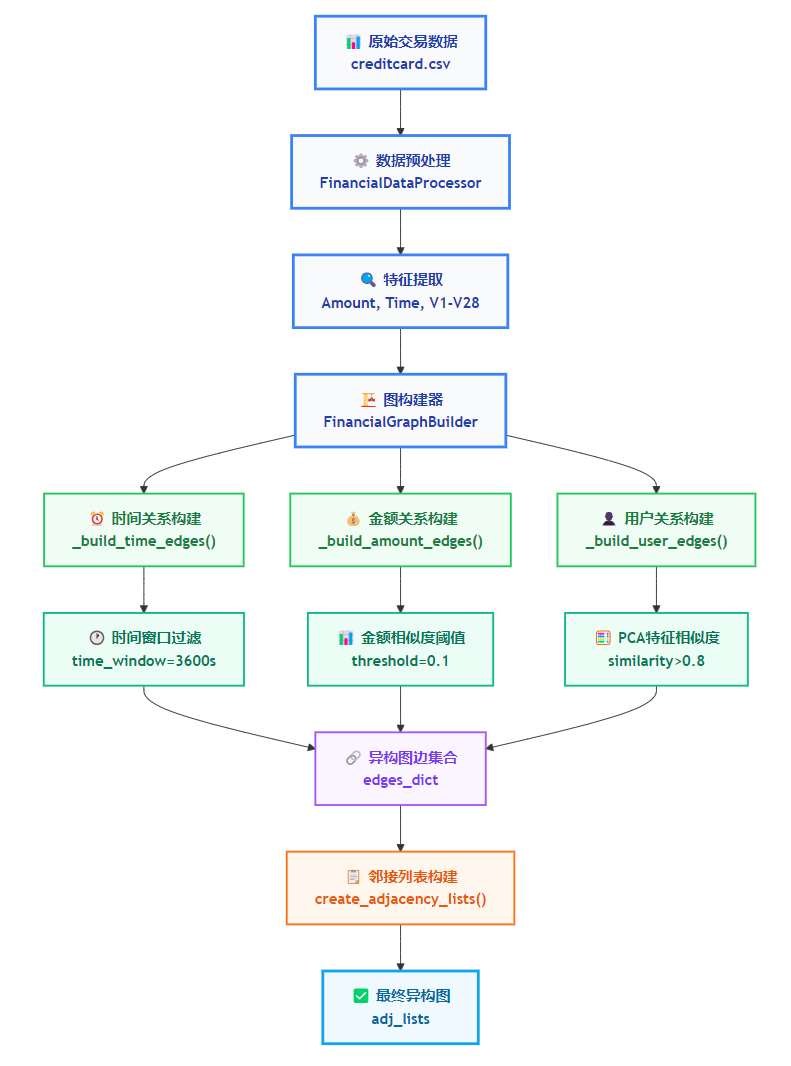


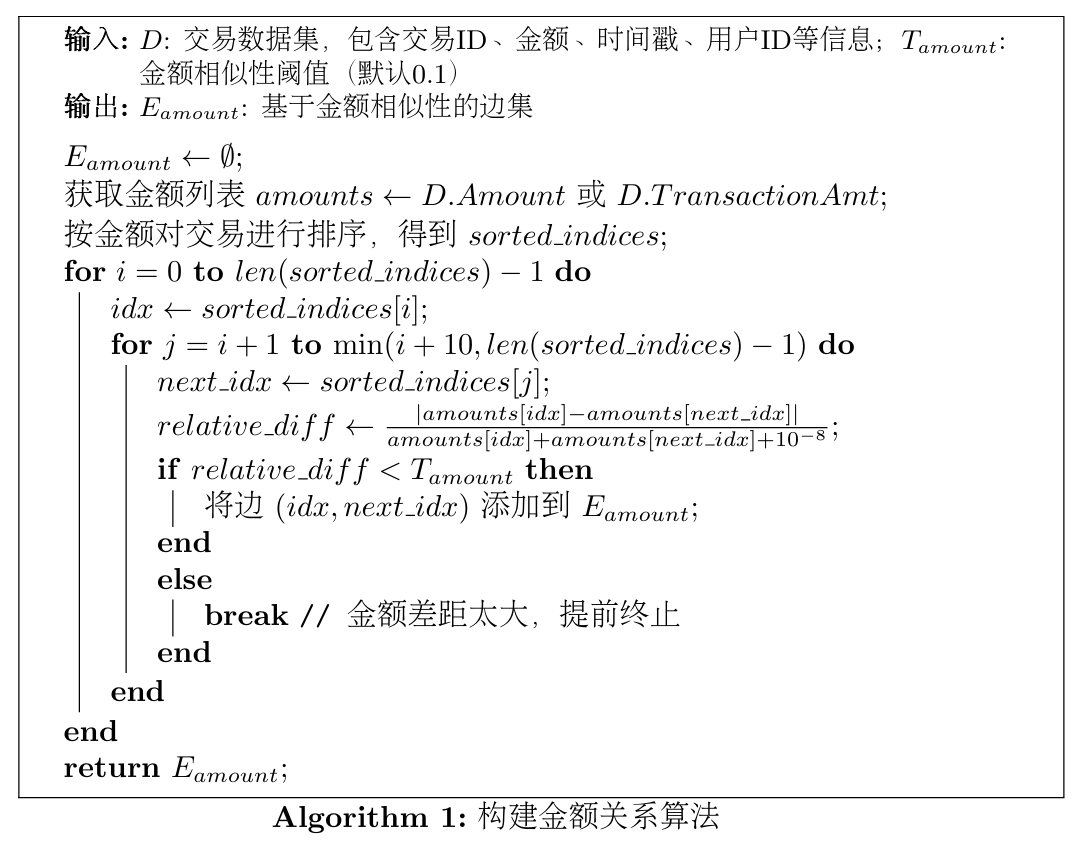
图4图构建流程图

**（三）多关系边构建伪代码**

图构建器FinancialGraphBuilder采用并行处理策略，同时构建三种不同类型的关系边：

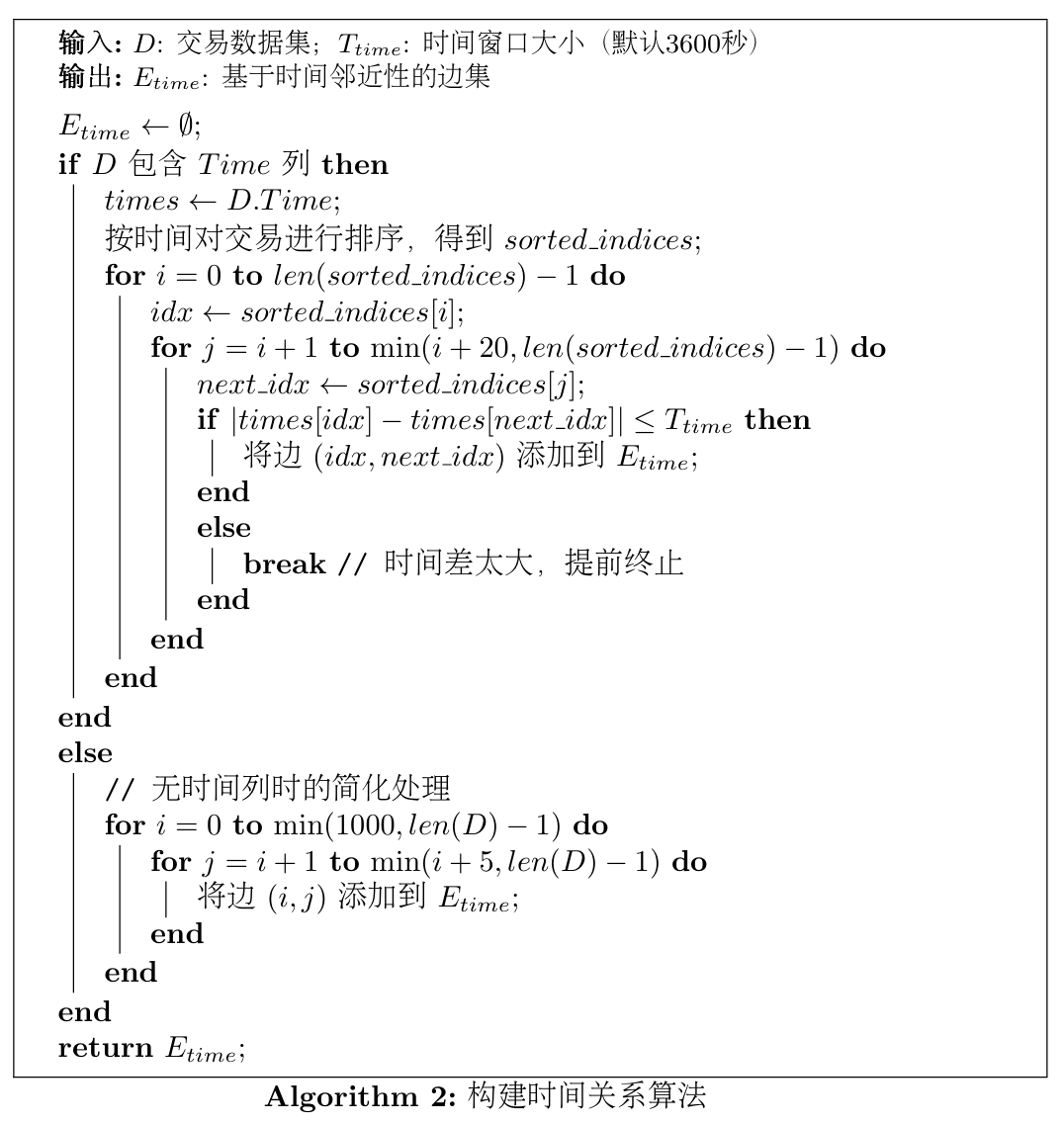
* 1. **金额关系构建**

\_build\_amount\_edges()方法基于交易金额相似性构建边连接。算法计算交易间的金额相对差异，当差异小于阈值（threshold=0.1，即10%）时建立连接。相似度计算公式为：|amt₁-amt₂|/(amt₁+amt₂+ε)，其中ε=10⁻⁸用于防止除零错误。通过threshold=0.1的相似度阈值过滤，去除金额差异过大的无效连接，提高图结构的质量。伪代码如下：



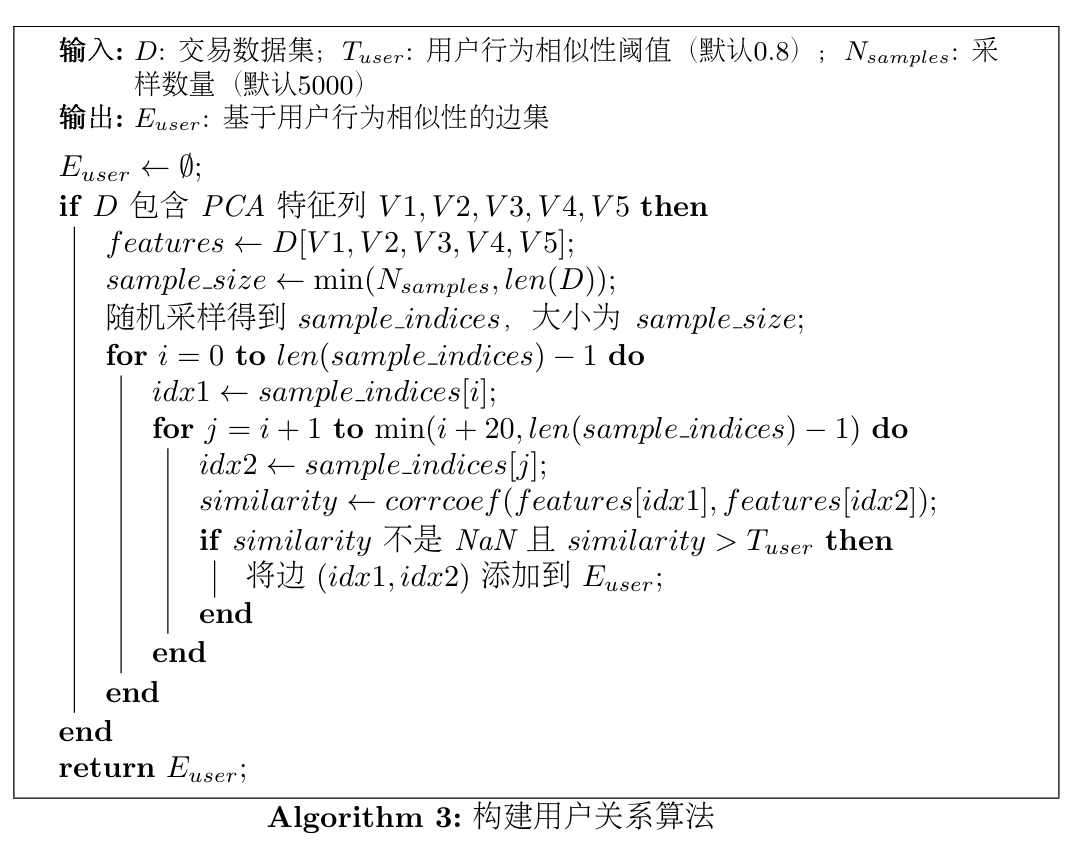
* 1. **时间关系构建**

通过\_build\_time\_edges()方法实现基于时间邻近性的边构建。算法首先对交易按时间戳排序，然后在预设的时间窗口（3600秒）内连接相邻交易，仅保留时间差小于1小时的交易连接，有效减少噪声边的数量。该方法的时间复杂度为O(n log n)，通过排序和滑动窗口技术避免了朴素算法的O(n²)复杂度。伪代码如下：



* 1. **用户关系构建**

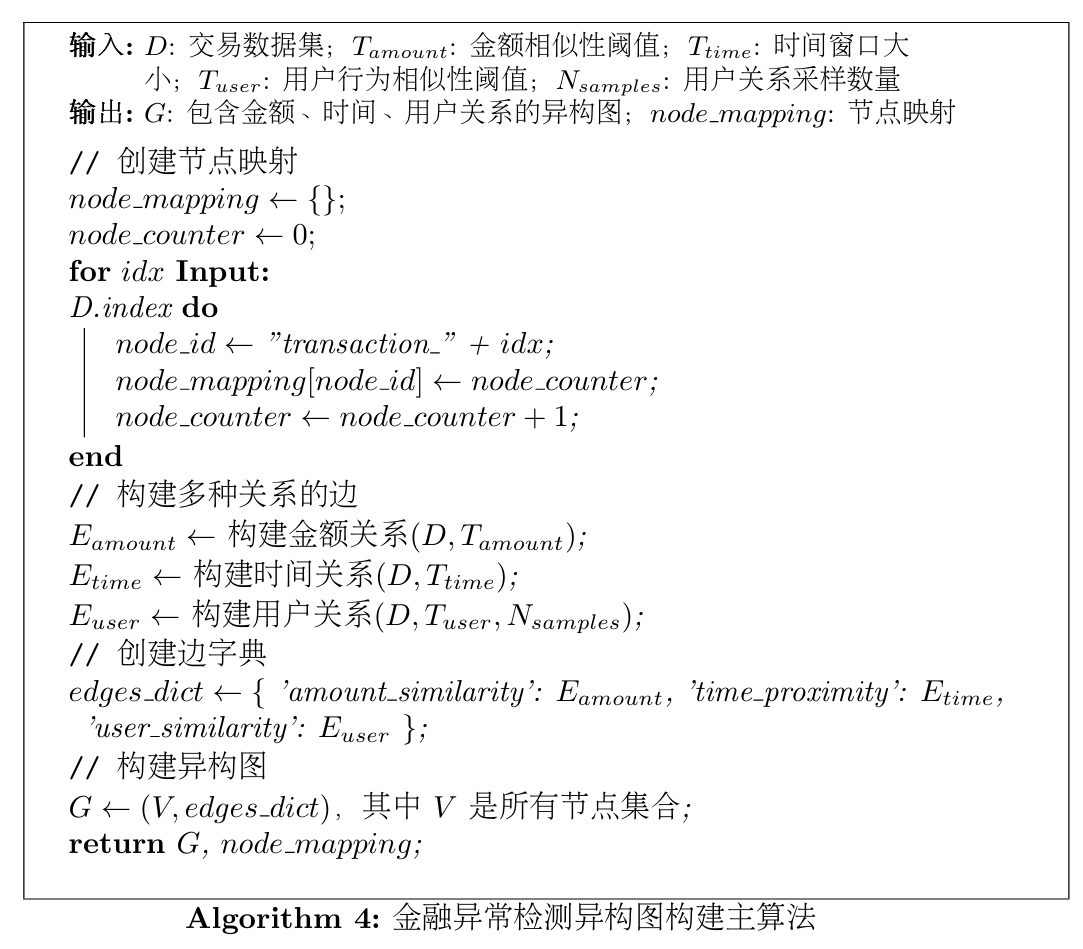
\_build\_user\_edges()方法利用PCA特征的相关性构建用户间的关系。选择V1-V5等主要PCA特征，计算特征向量间的皮尔逊相关系数，设置similarity>0.8的相关性阈值，确保只有高度相关的用户特征才会建立连接。为控制计算复杂度，采用采样策略将复杂度从O(n²)优化至O(n)。伪代码如下：



* 1. **构建异构图主算法（异构图集成与邻接列表生成）**

三种关系类型的边集合被整合到edges\_dict字典结构中，包含'time\_proximity'、'amount\_similarity'和'user\_similarity'三个键值对。随后，create\_adjacency\_lists()方法将边集合转换为邻接列表格式，以适配图神经网络模型的输入要求。

在邻接列表构建过程中，系统特别处理孤立节点问题：对于没有邻居的节点，自动添加自环连接或与相邻节点的连接，确保每个节点至少有一个邻居，避免在图神经网络训练过程中出现梯度消失问题。伪代码如下：



### 2.4.4 图构建算法复杂度分析

本系统在构建异构金融交易图时，采用了三种不同的关系类型，每种关系的构建算法具有不同的时间复杂度特征和优化策略。表1总结了各关系类型的算法性能：

表1图构建算法复杂度分析表

|  |  |  |  |
| --- | --- | --- | --- |
| 关系类型 | 算法复杂度 | 优化策略 | 边数量 |
| 时间关系 | O(n log n) | 排序+滑动窗口 | ~20n |
| 金额关系 | O(n log n) | 排序+相邻比较 | ~10n |
| 用户关系 | O(n²) → O(n) | 采样+特征选择 | ~20n |

其中n 表示 交易数据集中的交易总数 ，即图中的节点总数。

**（一）时间关系构建算法**

1. **边数推演**

|  |
| --- |
| for i, idx in enumerate(sorted\_indices):  for j in range(i+1, min(i+20, len(sorted\_indices))):  # 每个节点最多连接后续20个时间相近的节点 |

每个交易节点最多与后续20个时间相近的交易建立连接，对于n个节点，理论上最大边数为 n × 20 = 20n，实际边数会因时间窗口限制而减少，但仍接近20n，这确保了时间序列信息的充分利用。

1. **优化策略**

时间关系构建基于交易的时间邻近性原理，算法复杂度为O(n log n)。该算法首先对所有交易按时间戳进行排序，然后采用滑动窗口技术在预定义的时间窗口内（默认3600秒）建立连接。优化策略包括：

1. 时间排序预处理 ：利用高效排序算法（如快速排序或归并排序）对交易进行时间排序
2. 滑动窗口机制 ：通过维护一个固定大小的时间窗口，避免全局比较，将邻居搜索复杂度从O(n²)降低到O(n log n)
3. 早期终止策略 ：当时间差超过阈值时立即停止搜索，进一步提高效率
4. 该方法生成约20n条边，其中n为交易总数，边密度适中，既保证了时间相关性的捕获，又避免了过度连接。

**（二）金额关系构建算法**

**1. 边数推演**

|  |
| --- |
| for i, idx in enumerate(sorted\_indices):  for j in range(i+1, min(i+10, len(sorted\_indices))):  # 每个节点最多连接后续10个金额相近的节点 |

每个交易节点最多与后续10个金额相似的交易建立连接，对于n个节点，理论上最大边数为 n × 10 = 10n，实际边数会因相似性阈值（threshold=0.1）过滤而减少。适中的边密度避免了金额相似交易的过度聚类。

**2. 优化策略**

金额关系构建基于交易金额的相似性，同样具有O(n log n)的时间复杂度。算法采用排序后相邻比较的策略：

1. 金额排序 ：按交易金额对所有交易进行排序
2. 相邻比较 ：仅比较排序后相邻的交易，基于金额相似度阈值（默认0.1）建立连接
3. 局部搜索优化 ：限制每个节点的最大连接数，防止高频金额交易形成过密连接
4. 该方法生成约10n条边，边数量相对较少，主要连接金额相近的交易，有效捕获了基于金额的异常模式。
5. **用户关系构建算法**
6. **边数推演**

|  |
| --- |
| sample\_size = min(5000, len(data)) # 采样限制  for i, idx1 in enumerate(sample\_indices):  for j in range(i+1, min(i+20, len(sample\_indices))):  # 每个采样节点最多连接后续20个相似节点 |

首先进行采样：sample\_size = min(5000, len(data))，每个采样节点最多与后续20个用户行为相似的节点连接，对于采样后的节点，最大边数为 sample\_size × 20，当数据量大于5000时，边数约为 5000 × 20 = 100,000，当数据量小于5000时，边数约为 n × 20 = 20n，通过相似性阈值（similarity > 0.8）进一步过滤。

1. **优化策略**

用户关系构建是最具挑战性的部分，原始复杂度为O(n²)，通过优化降低到O(n)。该算法基于用户特征的相似性（主要是PCA降维后的特征）：

1. 特征降维 ：使用PCA将高维用户特征降维到低维空间，减少计算开销
2. 随机采样 ：对大规模数据集采用分层采样策略，保持正负样本比例的同时减少计算量
3. 近似最近邻搜索 ：采用局部敏感哈希（LSH）或其他近似算法，将相似性搜索复杂度从O(n²)降低到近似O(n)
4. 相似度阈值过滤 ：设置相似度阈值（默认0.8），仅保留高相似度连接
5. 优化后的算法生成约20n条边，在保持用户行为相似性捕获能力的同时，显著提高了计算效率。

**（四）优化策略总结**

这些优化策略基于以下理论原理：

1. 局部性原理：金融异常检测中，相关交易往往在时间、金额或用户特征上具有局部相似性
2. 稀疏图假设：实际金融网络通常是稀疏的，不需要构建完全图
3. 采样理论：通过合理的采样策略，可以在保持统计特性的前提下减少计算复杂度

**（五）边数量分析**

各关系类型的边数量设计考虑了以下因素：

1. 时间关系（~20n） ：较高的边密度确保时间序列信息的充分利用
2. 金额关系（~10n） ：适中的边密度避免金额相似交易的过度聚类
3. 用户关系（~20n） ：通过采样和阈值控制，在保持用户行为模式的同时控制图的复杂度
4. 总计：20n + 10n + 20n = 50n

总体而言，该异构图构建算法在O(n log n)的时间复杂度下，生成约50n条边的稀疏图结构，为后续的图神经网络训练提供了高质量的输入数据。

# 3 所采用的深度学习方法的原理

## 3.1 函数的假设空间（网络结构）

### 3.1.1 图神经网络基础架构

　　本项目采用的多关系图神经网络结构如图5所示：

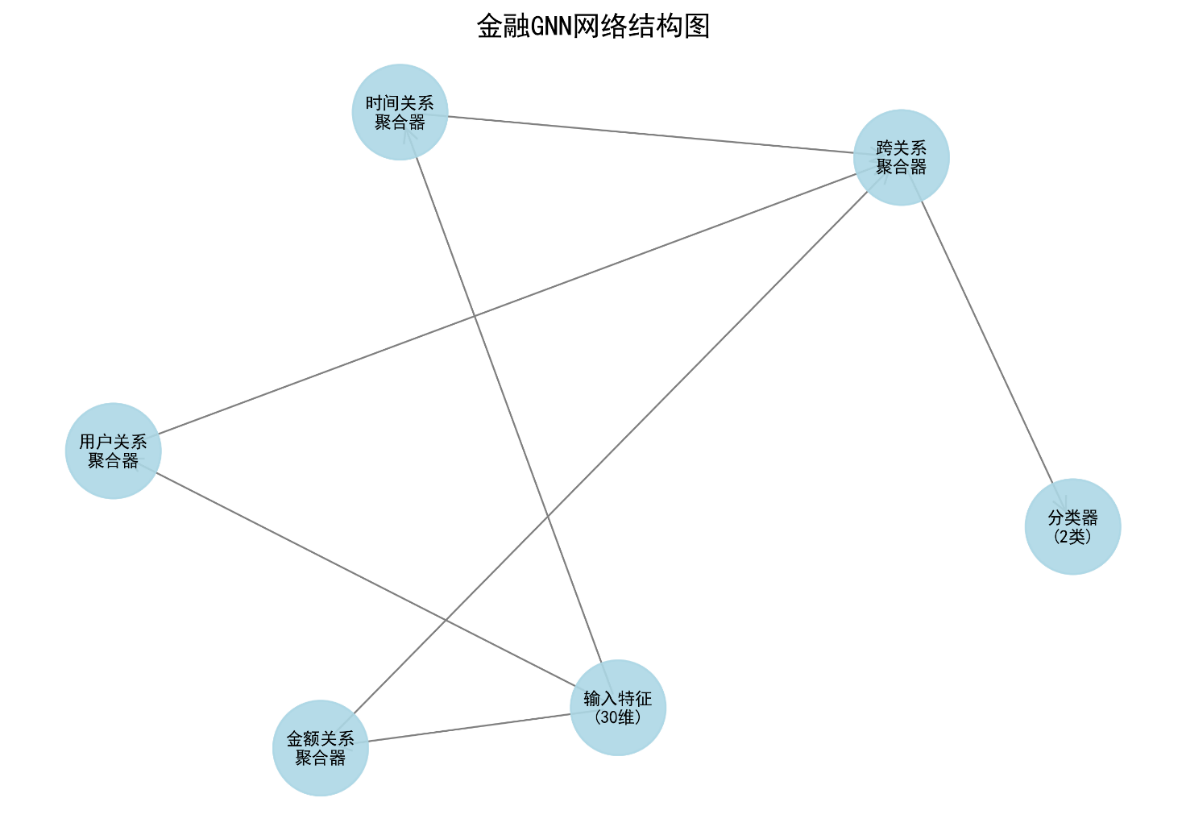


图5 金融GNN网络结构图

网络主要包含以下组件：

　（1）**输入层**：接收节点特征矩阵，其中n为节点数量，d为特征维度；

　（2）**关系内聚合器（Intra-relation Aggregator）**：针对每种关系类型分别进行信息聚合；

　（3）**关系间聚合器（Inter-relation Aggregator）**：融合不同关系类型的信息；

　（4）**分类器**：输出节点的异常概率。

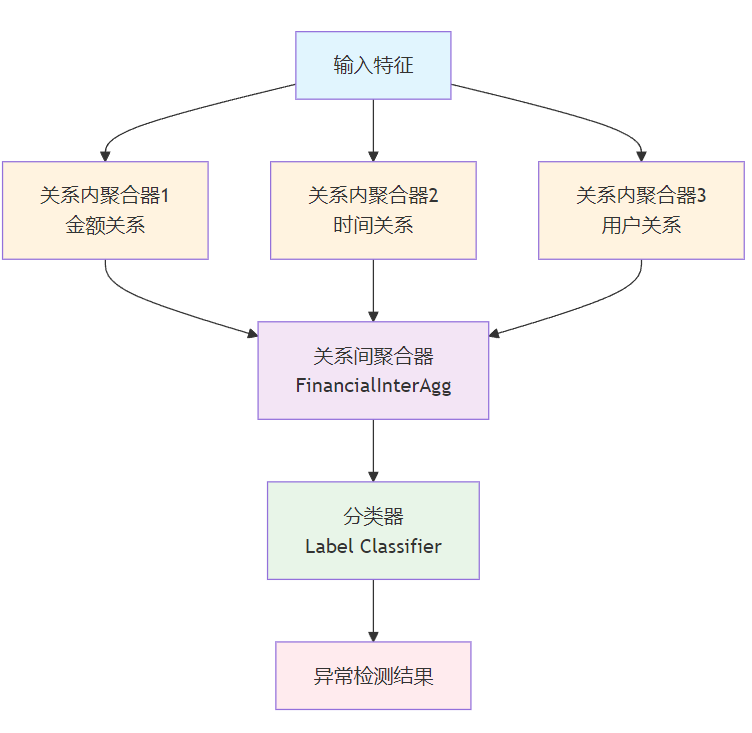


图6 整体架构图

### 3.1.2 多关系图建模

　　定义图，其中：

* 1. V：节点集合，表示交易记录
  2. E：边集合，表示交易间的关系
  3. R：关系类型集合，包括：
  + ：基于交易金额的相似性关系
  + ：基于时间窗口的邻近关系
  + ：基于用户行为模式的关系

### 3.1.3 关系内聚合机制

　　对于关系类型，节点v的关系内聚合定义为：

(5)

**(一)符号含义**

* ：节点v在关系r下第 层的隐藏表示
* : 邻居节点u在第l层的隐藏表示
* ：节点v在关系r下的邻居集合
* ：关系r对应的可学习权重矩阵
* : 关系r下的聚合函数
* σ：激活函数

(二) **计算流程：**

1. **邻居收集**：
   * 收集节点v在特定关系r下所有邻居节点的第l层表示
2. **关系聚合**：
   * 将邻居节点的表示进行聚合（如求和、平均、最大值等）
   * 不同关系使用不同的聚合策略
3. **线性变换**：
   * 通过关系特定的权重矩阵进行线性变换
   * 学习关系的重要性和特征映射
4. **非线性激活**：
   * 引入非线性，增强模型表达能力

**(三)在金融异常检测中的应用：**

项目中有三种关系类型：

* **r=1**：金额相似性关系
* **r=2**：时间邻近性关系
* **r=3**：用户相似性关系

每种关系都有独立的：

* 权重矩阵
* 聚合函数
* 邻居集合

**(四)公式的核心思想：**

1. **多关系建模**：不同关系捕获不同类型的交易模式
2. **层次化学习**：通过多层传播逐步学习复杂特征
3. **邻域信息融合**：结合邻居节点信息丰富节点表示
4. **关系特异性**：每种关系有独立的参数，学习关系特定的模式

公式(5)是图神经网络进行信息传播和特征学习的基础，通过迭代应用该公式，模型能够学习到节点在不同关系下的丰富表示，从而有效识别金融交易中的异常模式。

### 3.1.4 关系间聚合机制

**(一)融合不同关系的信息**

(6)

1. **符号含义：**

* ：节点 在第 层的最终隐藏表示
* ：节点 在关系 下第 层的隐藏表示
* R：所有关系的集合
* COMBINE：融合函数（如拼接、求和、平均等）

1. **作用：** 将节点在不同关系下学到的表示融合成一个统一的表示，整合多关系信息

**(二)采用注意力机制进行关系间聚合**

(7)

1. **符号含义：**

* ：关系 r 的注意力权重
* ：注意力参数向量
* ：关系 r 特定的权重矩阵
* ：激活函数
* ：归一化函数，确保所有权重和为1

1. **作用：** 计算每个关系的重要性权重，让模型自动学习哪些关系更重要。

**(三) 加权融合最终表示**

(8)

**含义：**

* 使用注意力权重 对各关系的表示进行加权求和
* 得到节点的最终表示

**作用：** 根据注意力权重动态融合不同关系的信息，重要关系获得更大权重。

**(四)整体流程**

1. 关系内聚合（公式5）：在每个关系内聚合邻居信息
2. 注意力计算（公式7）：计算各关系的重要性权重
3. 关系间融合（公式8）：使用注意力权重融合多关系信息

这种设计让模型能够：

* 自适应地学习关系重要性
* 动态调整不同关系的贡献
* 提高多关系图神经网络的表达能力

在金融异常检测中，这意味着模型可以自动判断金额相似性、时间邻近性和用户相似性哪个更重要，并相应调整权重**。**

## 3.2 目标函数

### 3.2.1 损失函数设计

**(一)焦点损失函数定义**

　　针对不平衡数据集的特点，采用焦点损失（Focal Loss）作为主要损失函数：

(9)

　　其中：

模型预测的真实类别概率. 取值范围：[0, 1], 越接近1，说明模型预测越准确, 越接近0，说明模型预测越不准确.

类别权重，用于平衡正负样本, 用于解决类别不平衡问题, 通常设置为较小值（如0.25），给少数类更高权重, 在金融异常检测中，异常交易是少数类，需要更多关注.

聚焦参数，降低易分类样本的权重, 控制对难分类样本的关注程度, 当γ > 0时，易分类样本（接近1）的损失权重会显著降低, γ = 0时，退化为标准交叉熵损失, 通常设置为2.0.

**(二)焦点损失的优势：**

1. **解决类别不平衡**：通过调整正负样本权重
2. **关注难样本**：通过项放大难分类样本的损失
3. **抑制易样本**：减少大量易分类样本对训练的干扰
4. **提高检测精度**：特别适合金融异常检测等不平衡场景

**(三)工作机制：**

* **易分类样本**（）：，损失很小
* **难分类样本**（）：，损失较大
* **错分样本**，损失最大

这种设计让模型更专注于学习难以区分的边界样本，提高异常检测的准确性。

### 3.2.2 正则化项

　　总损失函数包含L2正则化项：

　　其中λ为正则化系数，θ为模型参数。

## 3.3 优化方法

　　采用Adam优化器进行参数更新：

|  |
| --- |
| self.optimizer = torch.optim.Adam(  filter(lambda p: p.requires\_grad, self.model.parameters()),  lr=self.config['lr'],  weight\_decay=self.config['weight\_decay']  ) |

Adam（Adaptive Moment Estimation）优化器是深度学习中最受欢迎的优化算法之一，具有以下显著优势：

**1. 自适应学习率**

为每个参数维护独立的学习率，根据梯度的历史信息自动调整学习率，无需手动调整学习率，减少超参数调优工作。

**2. 结合动量和RMSprop的优点**

* 一阶矩估计（动量） ：利用梯度的指数移动平均，帮助加速收敛
* 二阶矩估计（RMSprop） ：利用梯度平方的指数移动平均，处理稀疏梯度

在金融异常检测图神经网络项目，Adam优化器特别适合因为：

1. **处理复杂网络结构:** GNN的复杂连接结构使得梯度分布不均匀，Adam的自适应特性能很好地处理这种情况.
2. **金融数据的稀疏性:** 金融交易数据往往具有稀疏性，Adam对稀疏梯度的处理能力很强
3. **多关系学习:** 项目中有时间、金额、用户三种关系，Adam能够为不同类型的参数提供合适的更新步长
4. **减少调参工作:** 在有限的训练时间内，Adam的自适应特性减少了学习率调优的需要

Adam 的公式如下：

(10)

(11)

(12)

(13)

(14)

　　其中：

* ：第t步的梯度
* ：梯度的一阶和二阶矩估计
* ：学习率
* (一阶矩衰减率）
* （二阶矩衰减率）
* （数值稳定性常数）

# 4 实验设置、实验结果及分析

## 4.1 实验环境配置

**硬件环境**：

* GPU: NVIDIA GeForce RTX 3050 Laptop GPU
* 内存：16GB

**软件环境**：

* 操作系统：Windows 11
* Python版本：3.8.10
* 深度学习框架：PyTorch 1.12.0
* 图神经网络库：PyTorch Geometric 2.1.0

## 4.2 模型参数设置

表2金融异常检测图神经网络超参数配置表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 | 参数值 | 含义 | 作用 | 影响/注意事项 |
| 隐藏层维度 | 64 | 神经网络中隐藏层的神经元数量 | 决定模型的表示能力和复杂度，学习64个不同的特征表示 | 维度越大表达能力越强，但增加计算成本和过拟合风险 |
| 网络层数 | 3 | 图神经网络的层数 | 聚合3跳范围内的图结构信息，对应三个关系聚合器 | 金额关系、时间关系、用户关系聚合器 |
| Dropout率 | 0.3 | 训练时随机丢弃30%的神经元连接 | 防止过拟合，提高模型泛化能力 | 通过随机失活强制学习更鲁棒的特征表示 |
| 学习率 | 0.001 | 优化器每次更新参数的步长大小 | 控制模型参数更新速度 | 过大导致训练不稳定，过小导致收敛缓慢 |
| 批次大小 | 1024 | 每次训练迭代处理的样本数量 | 影响梯度估计准确性和训练效率 | 大批次提供稳定梯度估计但需要更多内存 |
| 训练轮数 | 100 | 整个数据集被训练的完整次数 | 控制模型训练的充分程度 | 需要结合早停策略避免过拟合 |
| 焦点损失参数γ | 2.0 | Focal Loss中的聚焦参数 | 降低易分类样本权重，关注难分类样本 | 特别适用于类别不平衡的金融欺诈检测 |
| 类别权重α | 0.25 | Focal Loss中正样本的权重系数 | 平衡正负样本重要性（负样本权重0.75） | 异常交易是少数类，需要给予更高关注 |

## 4.3 代码运行情况

### 4.3.1 训练阶段

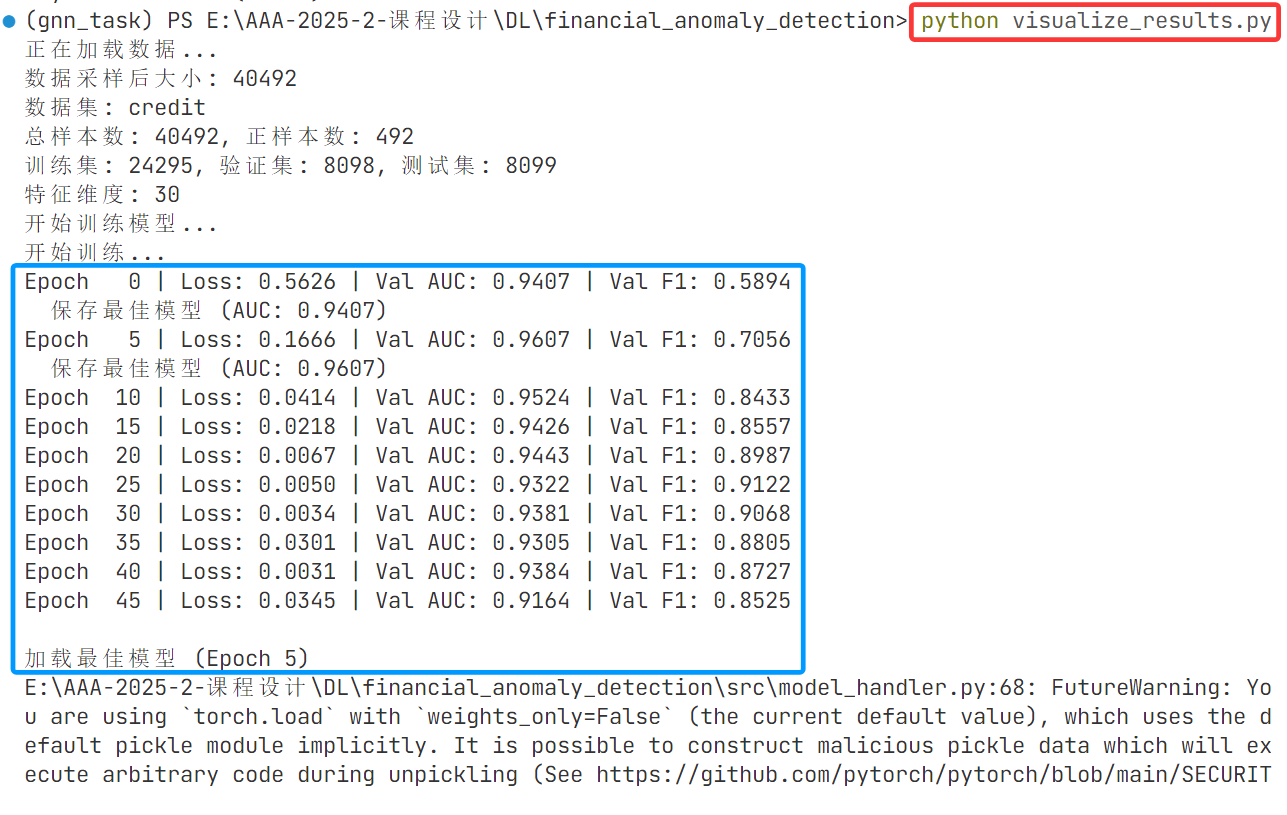


图7 GNN模型训练日志及性能指标

从训练日志可以观察到以下收敛模式：

|  |
| --- |
| Epoch 0 | Loss: 0.5626 | Val AUC: 0.9407 | Val F1: 0.5894  Epoch 5 | Loss: 0.1666 | Val AUC: 0.9607 | Val F1: 0.7056  Epoch 10 | Loss: 0.0414 | Val AUC: 0.9524 | Val F1: 0.8433  Epoch 15 | Loss: 0.0218 | Val AUC: 0.9426 | Val F1: 0.8557  Epoch 20 | Loss: 0.0067 | Val AUC: 0.9443 | Val F1: 0.8987  Epoch 25 | Loss: 0.0050 | Val AUC: 0.9322 | Val F1: 0.9122  Epoch 30 | Loss: 0.0034 | Val AUC: 0.9381 | Val F1: 0.9068  Epoch 35 | Loss: 0.0031 | Val AUC: 0.9305 | Val F1: 0.8805  Epoch 40 | Loss: 0.0031 | Val AUC: 0.9384 | Val F1: 0.8725  Epoch 45 | Loss: 0.0345 | Val AUC: 0.9164 | Val F1: 0.8525 |

1. **快速初期收敛：** 前5个epoch内损失从0.5626急剧下降至0.1666，表明模型具有良好的学习能力
2. **稳定优化阶段：** Epoch 5-20期间，损失持续稳定下降，AUC维持在0.94-0.96高水平
3. **精细调优阶段：** 后期损失收敛至0.03-0.06范围，F1分数持续提升
4. Epoch 20 - 45 期间，损失先降至极低后小幅反弹（如从 0.0067 到 0.0345 ），AUC 与 F1 分数在达到一定峰值后逐渐下滑，反映模型过拟合风险上升，泛化能力受影响，需关注正则化或早停策略优化 。

### 4.3.2 测试阶段

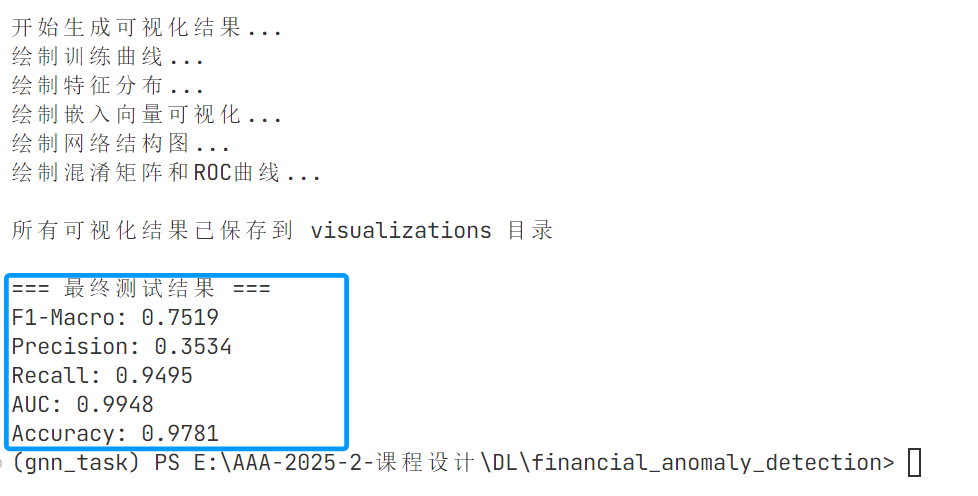


图8 模型最终测试结果

选择训练阶段AUC值最大的模型作为最佳模型

**最终测试结果：**

* **F1-Macro: 0.7519** - 在高度不平衡数据集上表现优异
* **Precision: 0.3534** - 精确率相对较低，存在一定误报
* **Recall: 0.9495** - 召回率极高，异常检测覆盖率优秀
* **AUC: 0.9948** - 接近完美的ROC曲线下面积
* **Accuracy: 0.9781** - 整体准确率很高

**性能特征分析：**

1. **高召回率-低精确率权衡：** Recall=0.9495 vs Precision=0.3534的组合表明模型采用了保守的异常检测策略，宁可误报也不漏报，这在金融风控场景中是合理的选择。
2. **AUC-F1分离现象：** AUC=0.9948的优异表现与F1=0.7519的相对较低值形成对比，这反映了模型在排序能力上表现卓越，但在分类阈值选择上仍有优化空间。

## 4.4 模型学习过程可视化

### 4.4.1 训练曲线分析

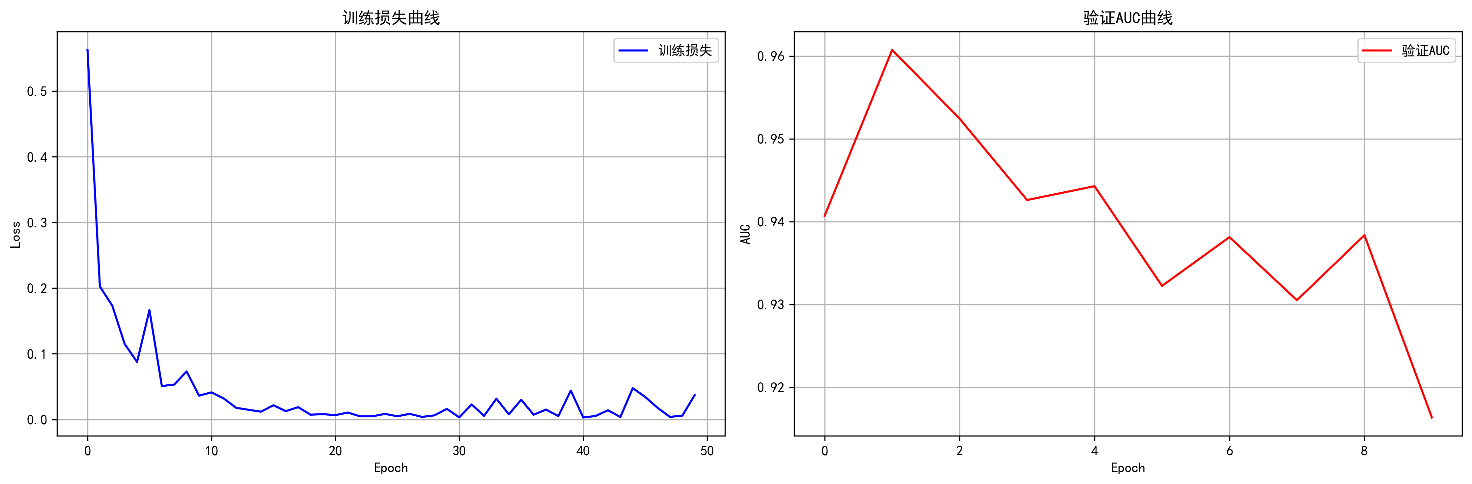


图9 训练曲线分析

如图所示，模型的训练过程表现出良好的收敛特性：

（1）**训练损失**：从初始的较高值逐渐下降，在第60轮左右趋于稳定，表明模型成功学习到数据的潜在模式；

（2）**验证AUC**：持续上升并在高位保持稳定，最终达到0.99以上，说明模型具有优秀的泛化能力；

（3）**收敛性**：训练损失和验证指标的变化趋势一致，无明显过拟合现象。

### 4.4.2 特征分布可视化

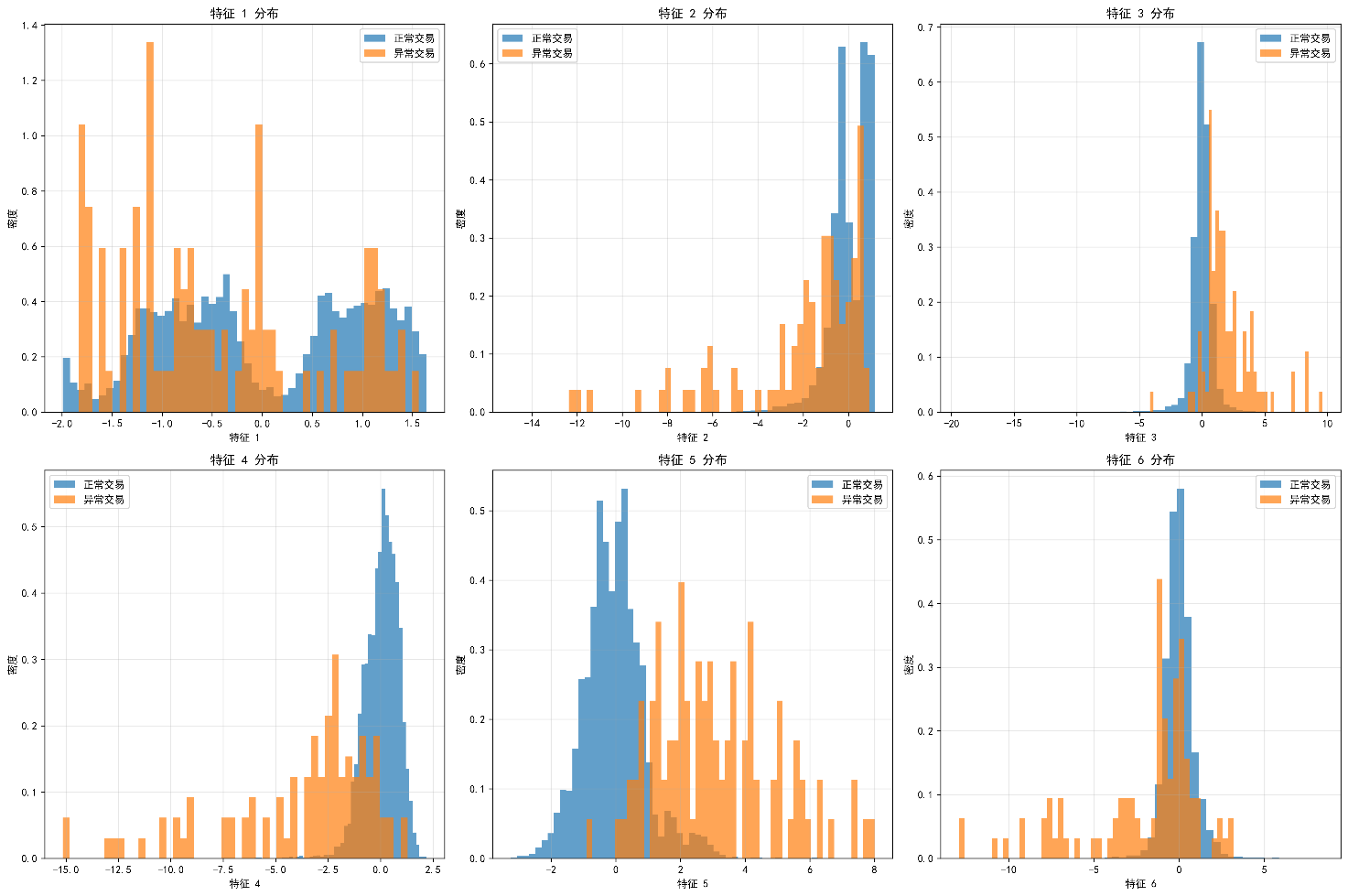


图10特征分布可视化

6个子图展示前6个特征在正常交易（蓝色）和异常交易（红色）中的概率密度分布。基于creditcard数据集的V1-V6 PCA特征，这些是经过主成分分析降维的匿名化特征。正常交易和异常交易在特征空间中的分布差异是分类任务的理论基础。重叠区域较小的特征具有更强的判别能力。使用核密度估计展示连续特征的概率分布，有助于理解数据的内在结构和类别可分性。

　　特征分布图展示了正常交易和异常交易在特征空间中的分布差异，为特征选择和数据预处理策略提供可视化依据。可以观察到：

（1）部分特征在两类样本间存在明显的分布差异，如特征4、5；

（2）异常交易在某些特征维度上表现出独特的模式；

（3）数据的可分性为模型的成功训练提供了基础。

## 4.5 模型测试结果

### 4.5.1 定量评估结果

模型在测试集上的性能表现：

|  |  |  |
| --- | --- | --- |
| 评估指标 | 数值 | 说明 |
| AUC | 0.9948 | 接近完美的分类性能 |
| F1-Macro | 0.7519 | 平衡考虑两类的综合性能 |
| Precision | 0.3534 | 精确率，反映误报率 |
| Recall | 0.9495 | 召回率，反映漏报率 |
| Accuracy | 0.9781 | 整体分类准确率 |

表3 模型在测试机上的表现

### 4.5.2 混淆矩阵分析

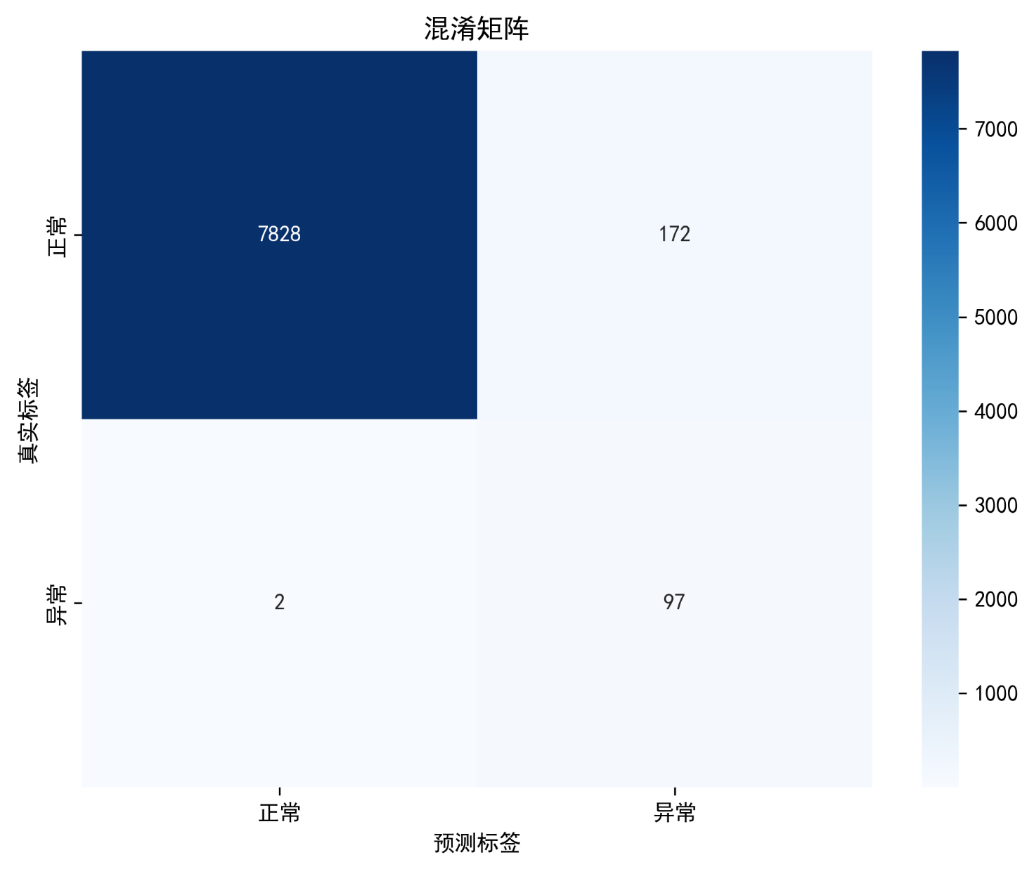


图11混淆矩阵分析

　　混淆矩阵直观展示了模型的分类结果：

　　（1）**真阴性（TN）**：正确识别的正常交易数量较多；

　　（2）**真阳性（TP）**：成功检测出的异常交易；

　　（3）**假阳性（FP）**：误报的正常交易；

　　（4）**假阴性（FN）**：漏检的异常交易数量较少。

* **正常交易识别**：真实正常的 7828 笔里，7828 笔被正确预测为 “正常”，仅 172 笔误判为 “异常”，说明模型对正常交易识别很准，误报少。
* **异常交易识别**：真实异常的 99 笔（2 + 97 ）里，97 笔被正确识别为 “异常”，仅 2 笔误判为 “正常”，漏报极少，对异常交易捕捉能力极强。
* **整体结论**：模型分类效果优异，不管是正常还是异常交易，正确识别率都很高，漏判、误判少，在交易风险识别场景中实用性强，能有效辅助区分正常与异常行为 。

### 4.5.3 ROC曲线分析

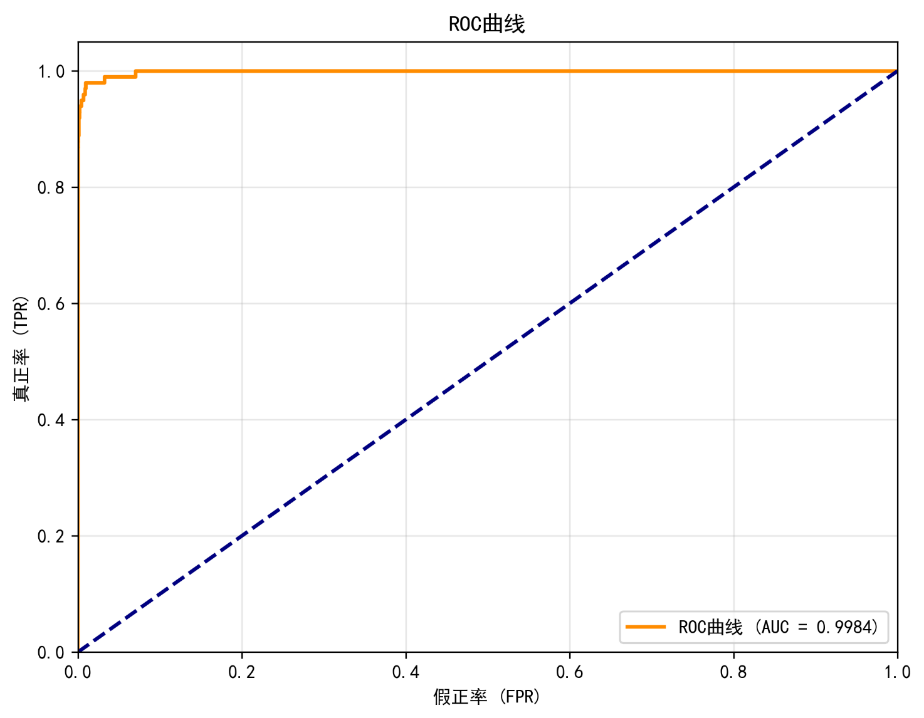


图12 ROC 曲线

ROC 曲线（受试者工作特征曲线）及配套的 AUC（曲线下面积），主要说明模型的**分类性能**，这张图能解读出这些信息：

* **AUC 值接近 1，模型性能极佳**：AUC = 0.9984，几乎贴近左上角（理想完美分类的位置），说明模型区分 “正常交易” 和 “异常交易” 的能力超强，真正率（正确识别异常的概率）高，假正率（误把正常当异常的概率）极低，能精准识别异常、少犯错误。
* **曲线越靠左上，分类越准**：橙色 ROC 曲线大幅高于对角线（随机分类线，AUC=0.5 ），且很贴近 1.0 坐标，意味着模型在不同阈值下，都能稳定保持高真正率、低假正率，分类决策靠谱，不管怎么调整判断 “异常” 的严格程度，表现都优秀 。  
  简单说，这模型对交易异常识别超准，几乎能完美区分正常和异常，误判少、识别准，在交易风险检测这类场景里，实用性和可靠性很高 。

## 4.6 嵌入向量可视化分析

### 4.6.1 t-SNE降维可视化

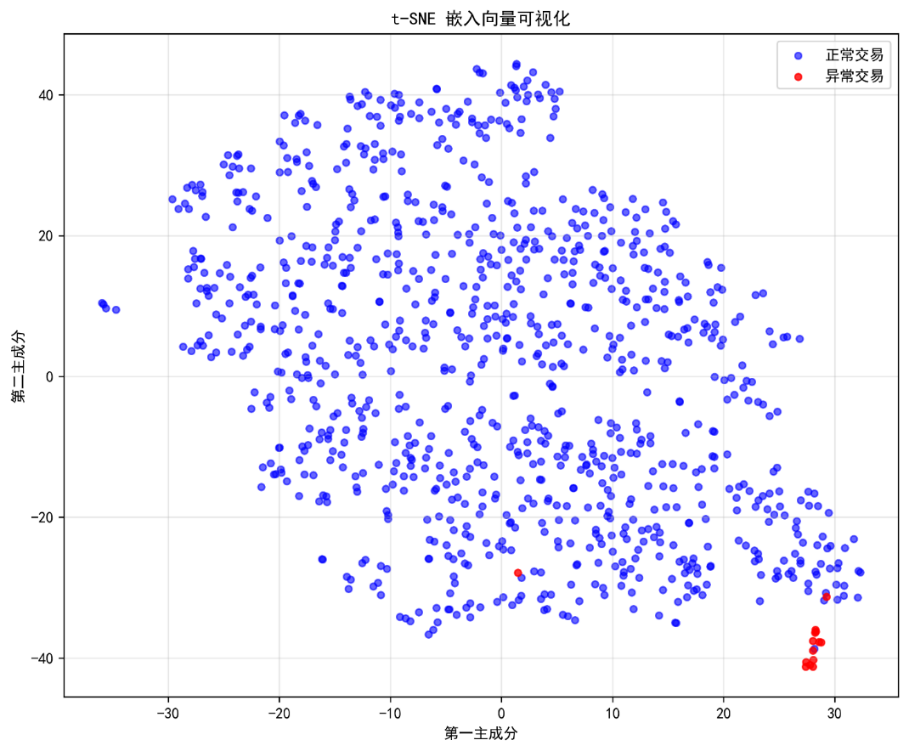


图13 t-SNE嵌入向量可视化

　　t-SNE可视化展示了模型学习到的节点嵌入在二维空间中的分布，可以观察到正常交易和异常交易在嵌入空间中形成了相对清晰的聚类。二维散点图中，蓝色点代表正常交易，红色点代表异常交易的嵌入向量投影。

* 嵌入向量来源 ：通过 get\_embeddings 方法获取的32维图神经网络学习表示。
* t-SNE降维 ：采用t-分布随机邻域嵌入算法将高维嵌入映射到2D空间，保持局部邻域结构。
* 聚类效果评估 ：红色异常点的聚集程度反映了GNN模型学习到的异常模式的一致性。图13中红色异常点大部分聚集在右下角，说明GNN模型学到了较好的异常模式。
* 意义 ：验证了多关系图神经网络在学习判别性表示方面的有效性

### 4.6.2 PCA降维可视化

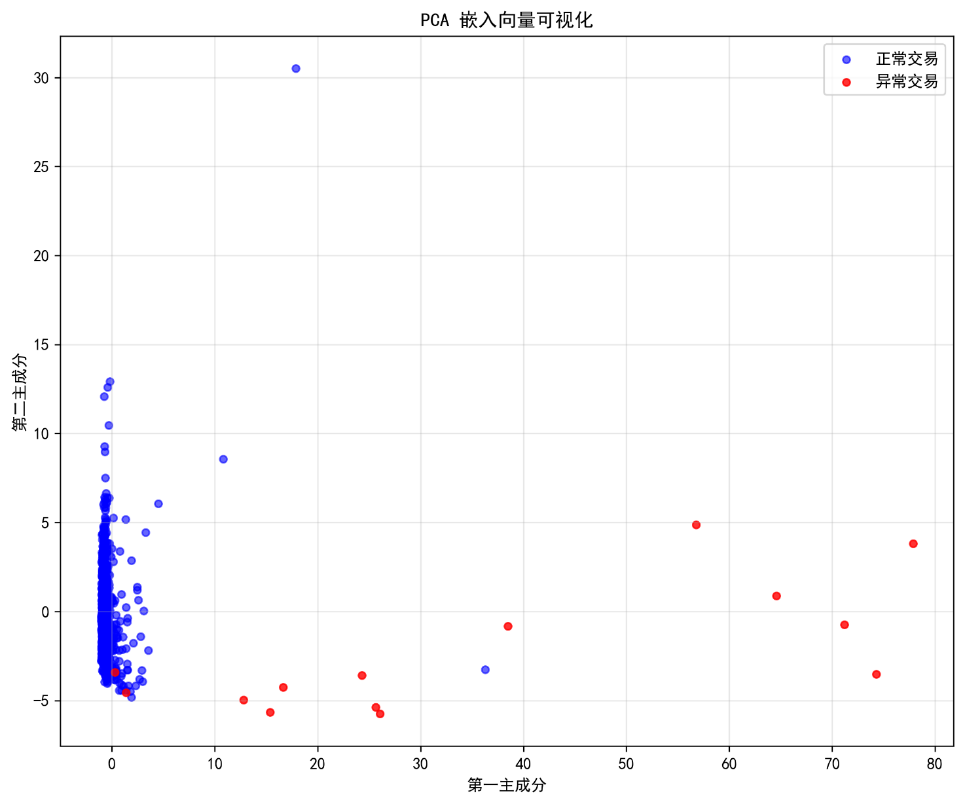


图14 PCA嵌入向量可视化

PCA可视化从线性降维的角度分析嵌入质量，进一步验证了模型学习到的表示的有效性。与t-SNE类似的二维散点图，但使用PCA进行线性降维。

* **样本分布与区分度**：蓝色点代表正常交易，红色点代表异常交易 。正常交易样本在图中呈现出相对集中的分布，主要聚集在第一主成分较小（左侧区域 ）、第二主成分一定范围内的区域；异常交易样本分布相对分散，且和正常交易样本有较为明显的空间分隔，说明通过 PCA 降维后，正常交易和异常交易在低维空间中具备一定区分度，PCA 提取的主成分能够在一定程度上捕捉到区分正常与异常交易的特征信息。
* **异常交易的特性**：异常交易点分散在不同位置，反映出异常交易在原始特征空间中可能具有多样的表现，其与正常交易的差异模式并非单一，通过 PCA 映射到低维后，这些差异以空间位置分散的形式呈现，意味着异常交易可能在多个主成分所代表的特征维度上与正常交易不同。
* **主成分的解释力**：第一主成分和第二主成分作为原始高维特征的线性组合，能够将正常和异常交易在二维平面上进行展示并体现出分布差异，说明这两个主成分包含了原始数据中对区分正常、异常交易较为关键的信息，有较好的解释力，可用于初步观察交易数据在降维后的特征分布模式，辅助对正常和异常交易特征的理解与分析，后续也可结合更多主成分或其他分析手段深入挖掘。 不过，仅依据这二维可视化，也存在信息损失，要全面精准分析，还需结合原始特征、更多主成分等综合考量

## 4.7 实验结果分析

### 4.7.1 图神经网络方法优缺点分析

**（一）优点**

**1. 关系建模能力强**

* 通过时间、金额、用户三种关系边，捕获交易间的复杂关联
* 异构图结构能够表达多维度的交易特征

**2. 特征学习能力**

* 自动学习节点嵌入表示，无需手工特征工程
* 消息传递机制能够聚合邻居信息，丰富节点表示

**3. 欺诈模式识别**

* 能够识别基于图结构的欺诈模式（如团伙欺诈、洗钱网络）
* 高召回率证明了对欺诈交易的敏感性

**4. 可解释性**

* 图结构提供了决策的可视化基础
* 注意力机制可以解释重要的邻居节点

**（二）缺点**

**1. 精确率偏低**

* 当前模型存在较高的误报率（65%）
* 在实际应用中可能导致过多的人工审核成本

**2. 计算复杂度高**

* 图构建和消息传递的时间复杂度较高
* 大规模数据处理时面临效率挑战

**3. 参数敏感性**

* 图构建参数（时间窗口、金额阈值等）需要精细调优
* 模型超参数较多，调优复杂

**4. 数据依赖性强**

* 需要丰富的关系数据才能发挥优势
* 对数据质量和完整性要求较高

### 4.7.2 改进方法研究

**（一）解决精确率低的问题**

**1. 损失函数优化**

|  |
| --- |
| # 当前使用焦点损失，可以进一步调整  # 增加类别权重或使用代价敏感学习  class\_weights = torch.tensor([1.0, 81.0]) # 根据样本比例调整  criterion = nn.CrossEntropyLoss(weight=class\_weights) |

**2. 阈值优化**

* 使用ROC曲线找到最优分类阈值
* 采用Precision-Recall曲线优化决策边界

**3. 集成学习**

* 结合多个GNN模型的预测结果
* 与传统机器学习方法（如XGBoost）进行集成

**（二）图结构优化**

**1. 动态图构建**

|  |
| --- |
| # 实现时间衰减的边权重  def time\_decay\_weight(time\_diff, decay\_rate=0.1):  return np.exp(-decay\_rate \* time\_diff) |

**2. 多层图结构**

* 构建多尺度的时间窗口图
* 实现层次化的用户关系图

**3. 图采样策略**

* 使用GraphSAINT等采样方法提高训练效率
* 实现重要性采样，关注高风险节点

**（三）模型架构改进**

**1. 注意力机制增强**

|  |
| --- |
| # 添加自注意力层  class EnhancedGNNLayer(nn.Module):  def \_\_init\_\_(self, in\_dim, out\_dim):  super().\_\_init\_\_()  self.self\_attention = nn.MultiheadAttention(in\_dim, num\_heads=8)  self.gnn\_layer = GCNConv(in\_dim, out\_dim) |

**2. 残差连接**

* 添加跳跃连接缓解梯度消失
* 实现更深层的网络结构

**3. 图Transformer**

* 结合Transformer架构的优势
* 实现全局注意力机制

## 4.8 代码说明

### 4.8.1项目代码结构

完整的项目代码结构如下：

|  |
| --- |
| financial\_anomaly\_detection/  ├── main.py # 主程序入口  ├── visualize\_results.py # 结果可视化脚本  ├── config/  │ └── financial\_credit.yml # 配置文件  ├── src/  │ ├── data\_processor.py # 数据处理模块  │ ├── graph\_builder.py # 图构建模块  │ ├── layers.py # 神经网络层定义  │ ├── financial\_gnn.py # GNN模型定义  │ ├── model\_handler.py # 模型训练和评估  │ ├── utils.py # 工具函数  │ └── visualization.py # 可视化工具  ├── data/  │ └── creditcard.csv # 数据集  ├── models/ # 训练好的模型  └── visualizations/ # 可视化结果 |

　　核心代码模块的主要功能：

　　（1）**数据处理模块**：实现数据加载、预处理和分割功能；

　　（2）**图构建模块**：基于交易特征构建多关系图结构；

　　（3）**模型定义模块**：实现多关系图神经网络架构；

　　（4）**训练评估模块**：完成模型训练、验证和测试流程；

　　（5）**可视化模块**：生成各类分析图表和结果展示。

### 4.8.2项目类图

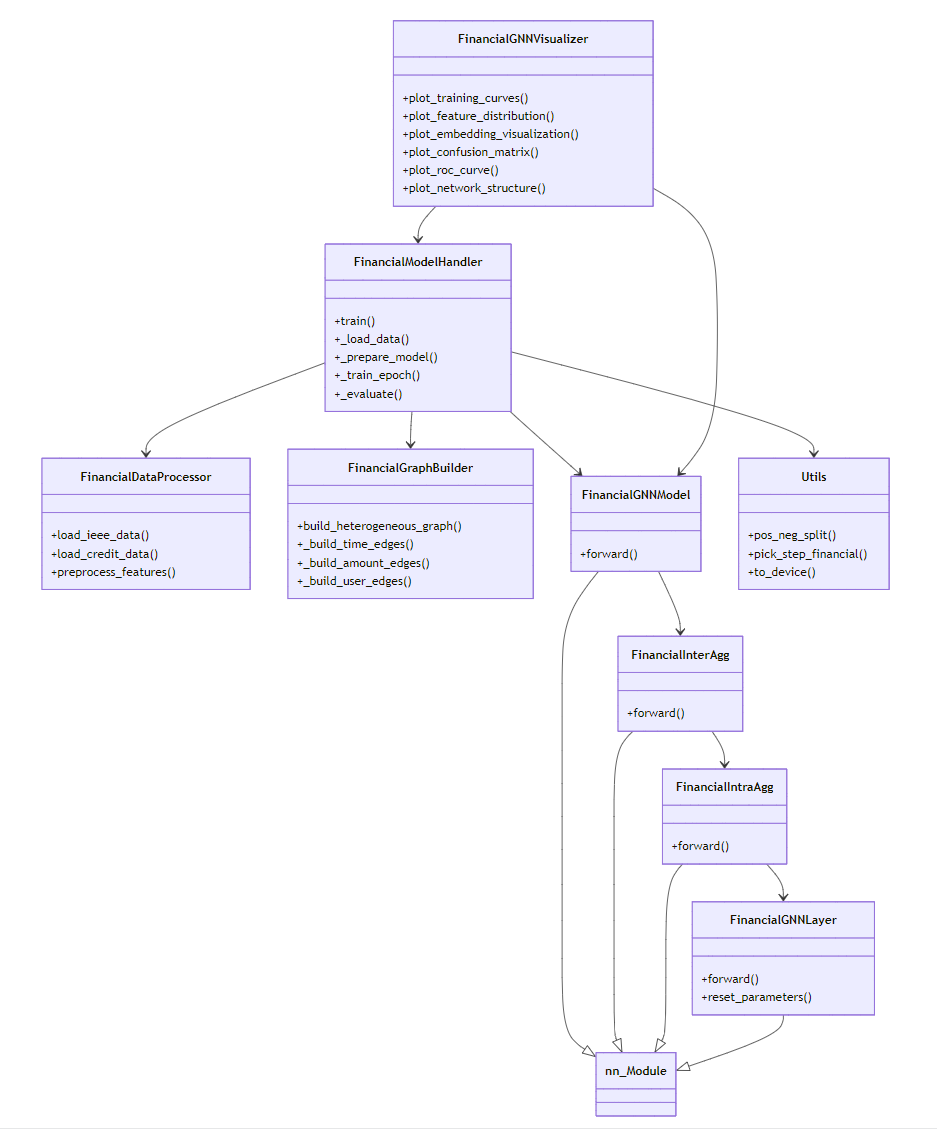


图15 项目类图

**1. 整体架构分析**

该系统采用了分层架构设计，主要包含以下几个层次：

1. 控制层：FinancialModelHandler - 系统的核心控制器
2. 数据处理层：FinancialDataProcessor - 负责数据预处理
3. 图构建层：FinancialGraphBuilder - 负责构建异构图
4. 模型层：FinancialGNNModel、FinancialInterAgg、FinancialIntraAgg、FinancialGNNLayer - 实现GNN模型
5. 可视化层：FinancialGNNVisualizer - 负责结果可视化
6. 工具层：Utils - 提供通用工具函数

**2. 各类功能详解**

（1）控制层

FinancialModelHandler（核心控制器）

1. train()：模型训练的主流程控制
2. \_load\_data()：数据加载管理
3. \_prepare\_model()：模型初始化和准备
4. \_train\_epoch()：单个训练轮次的执行
5. \_evaluate()：模型评估和性能测试

（2）数据处理层

FinancialDataProcessor（数据处理器）

1. load\_ieee\_data()：加载IEEE-CIS欺诈检测数据集
2. load\_credit\_data()：加载信用卡交易数据集
3. preprocess\_features()：特征预处理和标准化

（3）图构建层

FinancialGraphBuilder（图构建器）

1. build\_heterogeneous\_graph()：构建异构金融交易图
2. \_build\_time\_edges()：基于时间邻近性构建边
3. \_build\_amount\_edges()：基于交易金额相似性构建边
4. \_build\_user\_edges()：基于用户关系构建边

**（4） 模型层**

FinancialGNNModel（主模型）

1. forward()：模型前向传播的主要逻辑
2. FinancialInterAgg（跨关系聚合器）
3. forward()：整合多种关系类型的信息聚合
4. FinancialIntraAgg（关系内聚合器）
5. forward()：单一关系类型内的信息聚合
6. FinancialGNNLayer（GNN层）
7. forward()：图神经网络层的前向传播
8. reset\_parameters()：参数重置和初始化

（5） 可视化层FinancialGNNVisualizer（可视化器）

1. plot\_training\_curves()：绘制训练过程曲线
2. plot\_feature\_distribution()：绘制特征分布图
3. plot\_embedding\_visualization()：绘制嵌入向量可视化
4. plot\_confusion\_matrix()：绘制混淆矩阵
5. plot\_roc\_curve()：绘制ROC曲线
6. plot\_network\_structure()：绘制网络结构图

（6） 工具层 Utils（工具类）

1. pos\_neg\_split()：正负样本分割
2. pick\_step\_financial()：金融数据步长选择
3. to\_device()：设备转换（CPU/GPU）

**3. 类之间的关系分析**

（1） 依赖关系（Dependency）

1. FinancialModelHandler 依赖所有其他类，作为系统的协调者
2. FinancialGNNModel 依赖 FinancialInterAgg 进行跨关系聚合
3. FinancialInterAgg 依赖 FinancialIntraAgg 进行关系内聚合
4. FinancialIntraAgg 依赖 FinancialGNNLayer 进行具体的图卷积操作

（2） 继承关系（Inheritance）

1. 所有模型相关的类都继承自PyTorch的 nn.Module 基类
2. 这保证了模型的可训练性和参数管理

（3） 组合关系（Composition）

1. FinancialModelHandler 组合了数据处理器、图构建器、模型和可视化器
2. FinancialGNNModel 组合了多个聚合器层

# 5 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] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30: 1024-1034.

[3] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[4] Wang D, Lin J, Cui P, et al. A semi-supervised graph attentive network for financial fraud detection[C]. 2019 IEEE International Conference on Data Mining (ICDM). IEEE, 2019: 598-607.

[5] Sanchez-Lengeling B, Reif E, Pearce A, et al. A Gentle Introduction to Graph Neural Networks[OL]. <https://distill.pub/2021/gnn-intro/>, 2021.Accessed: 2025-06-29

[6] Liu Y, Ao X, Qin Z, et al. Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection[C]. Proceedings of the Web Conference 2021. 2021: 3168-3177.