基基于图神经网络的文本分类

神经网络与深度学习课程实验报告

班级：110

姓名：耿翊中

学号：2021213382

2023 年 11 月

目录

1. 实验要求

尝试复现 GCN(图卷积神经网络)和 GAT（图注意力网络）模型，从（20NG、R8、R52、Ohsumed、MR）至少选择 2 个数据集为训练数据，训练文本分类器并进行测试。要求：代码需有注释，最终提交代码（jupyter notebook 文件）和模型，并撰写报告，包含模型原理、实验设置、实验结果分析（例如分类结果、损失图等）等。

附加题 1.（5 分）在上述基础上，进一步复现 CNN、LSTM 等，与上述图神经网络方法比较的差异，最终提交代码、模型，并撰写报告。具体要求同上。
附加题 2.（5 分）思考如何利用异构图神经网络对包含有文本、图像、元数据信息（作者、发表期刊会议等）的多模态异构文档进行分类，通过调研目前存在的方法和策略，设计相应的解决方法，并分析方法的可行性以及创新性。（注：本题目不需要代码实现，要求有新意，并撰写报告。）

2. 模型原理

2.1 GCN：图卷积网络

2.1.1 GCN 结构

GCN 使用图结构数据，其中节点通过边相连。每个节点的特征信息通过卷积层更新，卷积层考虑了节点的邻居信息。

实现：

GCNConv 是 GCN 的基础卷积层。

模型包含两个 GCN 卷积层，第一个将输入特征转换为 8 维特征，第二个将这些特征转换为类别数对应的维度。

使用 ReLU 激活函数和 Dropout 防止过拟合。

```
class GCN(torch.nn.Module):
    def __init__(self, num_features, num_classes):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(num_features, 8)
        self.conv2 = GCNConv(8, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index

        x = F.relu(self.conv1(x, edge_index))
        x = F.dropout(x, training=self.training)
        x = self.conv2(x, edge_index)

        # print("Output shape:", x.shape)

        return F.log_softmax(x, dim=1)
```

2.1.2 GCN 原理

图的表示：在 GCN 中，图被表示为节点的特征矩阵和邻接矩阵。节点特征矩阵 X 包含了每个节点的特征向量，而邻接矩阵 A 描述了节点之间的连接关系。

卷积操作：GCN 的核心是图卷积操作。在传统的卷积神经网络中，卷积层通过在数据的局部区域上滑动过滤器来提取特征。而在 GCN 中，卷积操作被修改为操作在图的节点上，通过聚合一个节点及其邻居的特征信息来更新该节点的特征。

层级结构：GCN 通常包含多个图卷积层。每一层都会根据邻居节点的特征更新每个节点的特征表示。通过堆叠多层，模型可以捕获更高层次的图结构信息。

2.1.3 GCN 实现文本分类原理

文本预处理和特征提取：

文本首先经过清洗，去除短词和非字母字符，并进行词形还原。

使用 TF-IDF 方法将文本转换为数值型特征向量，构成特征矩阵 X。

构建图结构：

利用文本特征计算余弦相似度，构建邻接矩阵 A。相似度高于某个阈值的文本之间会在图中建立连接。

生成的邻接矩阵 A 反映了文本之间的相似关系。

GCN 模型构建：

模型中定义了两个 GCN 卷积层。第一个卷积层 GCNConv 将特征矩阵 X 中的每个节点（文本）转换为一个中间表示，第二个卷积层进一步将这些表示转换为用于分类的输出。

使用 ReLU 激活函数和 Dropout 来增加非线性和防止过拟合。

训练与测试：

在训练过程中，使用交叉熵损失函数（NLLLoss）来优化模型，使其能够正确分类文本。

在测试过程中，模型的性能通过准确率来评估。

2.2 GAT：图注意力网络

2.2.1 GAT 结构

GAT 同样适用于图结构数据，但在聚合邻居信息时使用注意力机制，使模型能够学习到不同邻居的重要性。

实现：

GATConv 是 GAT 的基础卷积层。

第一层 GAT 卷积层有 8 个注意力头，第二层有 1 个。

使用 ELU 激活函数和 Dropout。

```
class GAT(torch.nn.Module):
    def __init__(self, num_features, num_classes):
        super(GAT, self).__init__()
        self.conv1 = GATConv(num_features, 8, heads=8, dropout=0.6)
        # 如果您想要多层 GAT，可以在这里添加更多层
        self.conv2 = GATConv(8 * 8, num_classes, heads=1, concat=False, dropout=0.6)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index

        x = F.dropout(x, p=0.6, training=self.training)
        x = F.elu(self.conv1(x, edge_index))
        x = F.dropout(x, p=0.6, training=self.training)
        x = self.conv2(x, edge_index)

        return F.log_softmax(x, dim=1)
```

2.2.2 GAT 原理

图注意力网络（Graph Attention Network, GAT）是一种图神经网络（GNN），它通过引入注意力机制来改进图卷积网络（GCN）。GAT 能够动态地分配不同的权重到一个节点的邻居节点上，这允许模型更有效地聚合邻居的特征信息。

注意力机制：

在 GAT 中，每个节点通过学习到的注意力权重来聚合其邻居的特征信息。这意味着模型可以学习到每个邻居节点的重要性，并据此调整它们在特征聚合过程中的贡献度。

多头注意力：

GAT 通常采用多头注意力机制，这意味着它会并行地进行多个独立的注意力计算，并将结果合并。这样可以稳定学习过程并提高模型的表达能力。

特征更新：

类似于 GCN，GAT 在每一层都会更新节点的特征表示，但更新过程是基于注意力机制，而非简单的邻居平均。

2.2.3 GAT 实现文本分类原理

文本预处理和特征提取：

与 GCN 相同，首先对文本进行清洗和 TF-IDF 特征提取，构成特征矩阵 X。

构建图结构：

通过计算文本特征之间的余弦相似度来构建邻接矩阵 A。这个邻接矩阵反映了文本之间的相似关系。

GAT 模型构建：

在 GAT 类中定义了两个 GATConv 层。第一个层具有多个注意力头（在代码中为 8 个），用于从每个文本的特征中学习不同的表示。第二个层进一步将这些表示综合起来，形成最终的分类结果。

使用 ELU 激活函数和 Dropout 来增加非线性和防止过拟合。

训练与测试：

训练过程中，使用负对数似然损失（NLLLoss）来优化模型。

在测试过程中，评估模型在不同数据集（训练集、验证集、测试集）上的表现。

3. 实验设置

3.1 数据准备

3.1.1 数据集选择

选择 R8 和 R52 数据集

R8 和 R52 是两个常用的文本分类数据集，它们都是路透社 21578 数据集的子集。路透社 21578 数据集是一个包含 21,578 篇新闻文章的数据集，每篇文章都有一个或多个类别标签。R8 和 R52 是根据文章的主题来划分的，R8 有 8 个类别，R52 有 52 个类别。R8 和 R52 的数据集统计如下：

数据集	类别数	训练集大小	测试集大小	词汇量
R8	8	5,485	2,189	7,674
R52	52	6,532	2,568	9,060

数据集来源于 yao8839836/text_gcn: Graph Convolutional Networks for Text Classification. AAAI 2019 (github.com)

3.1.2 预处理和特征提取

对文本进行清洗、词形还原，并使用 TF-IDF 方法提取特征。

文本清洗（clean_text 函数）：

将文本转换为小写。移除长度小于等于 2 的单词。移除非字母字符。使用 WordNet 词形还原器对单词进行词形还原。移除停用词。

加载数据集（load_dataset 函数）：

从文件中加载数据。对每一行进行文本清洗。返回清洗后的文本列表。

构建特征矩阵（build_feature_matrix 函数）：

使用 TF-IDF 向量化器，提取文本的重要特征。选择最重要的 1000 个特征。返回文本特征矩阵。

构建邻接矩阵（build_adjacency_matrix2 函数）：

使用余弦相似度计算文本特征之间的相似度。基于相似度大于 0.5 的条件构建二值的邻接矩阵。对角线元素设为 0，避免自环。使用稀疏矩阵表示邻接矩阵。

获取节点特征（get_feature_matrix 函数）：

根据词汇表和词向量构建节点特征矩阵。

加载标签（load_labels 函数）：

从文件中加载标签。创建从标签名到整数的映射。将标签转换为整数。

3.1.3 划分数据集

将数据集划分为训练集、验证集和测试集。确保这三个子集的划分在对比 GCN 和 GAT 时是相同的。

3.2 模型设置和训练

3.2.1 模型设置

统一参数：为了保证实验的公平性，确保 GCN 和 GAT 使用相同的参数

学习率：设定一个适当的学习率， 0.1 和 0。01。

迭代次数（Epochs）：设置一个足够的迭代次数，如 200 或 300，以确保模型充分训练。

Dropout 率：设置为 0.5。

隐藏层维数：确保 GCN 和 GAT 的隐藏层具有相同的维数。

环境设置：确保所有实验在相同的硬件和软件环境下进行。

3.2.2 训练过程

初始化模型：分别初始化 GCN 和 GAT 模型。

优化器：选择合适的优化器，Adam。

损失函数：使用适合分类任务的损失函数，这里选择交叉熵损失（cross-entropy loss）。

训练循环：对于每个 Epoch，进行以下步骤：

在训练集上运行模型，计算损失并进行梯度下降。

在每个 Epoch 结束后，记录训练损失和准确率，记录准确率，精准率，召回率。

3.3 评估比较

3.3.1 损失和准确率曲线

绘制模型在训练过程中的损失和准确率曲线，比较两种模型的收敛速度和稳定性。

损失曲线：

损失曲线反映了模型在训练过程中的优化程度。随着训练的进行，损失应该逐渐减小。比较两种模型的损失曲线，可以了解它们在学习任务上的表现。如果一个模型的损失下降更快，可能表示它更快地学到了数据的特征。

准确率曲线：

准确率曲线显示模型在每个训练步骤后的准确率。准确率是分类任务中常用的评估指标，表示正确分类的样本比例。比较两种模型的准确率曲线可以帮助确定它们的性能差异。更高的准确率通常意味着模型学到了更好的特征表示。

比较收敛速度和稳定性：

收敛速度：观察损失曲线的降低速度。较快的降低可能表示模型更快地适应数据。

稳定性：观察曲线的平稳程度。较平稳的曲线表示模型在训练过程中更稳定，不容易受到噪声的影响。

3.3.2 性能指标

使用精准率，召回率和 F1 分数评估两种模型在测试集上的性能。

精准率（Precision）：

精准率表示被分类为正例的样本中有多少确实是正例。计算公式为

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

模型的精准率越高，表示模型在正例的分类上越准确。

召回率（Recall）：

召回率表示实际为正例的样本中有多少被正确地分类为正例。计算公式为

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

模型的召回率越高，表示模型更好地捕捉到了正例。

F1 分数：

F1 分数是精准率和召回率的调和平均数，综合考虑了这两个指标。计算公式为

$$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 分数在模型需要同时考虑精准率和召回率时提供了一个综合性的评价。

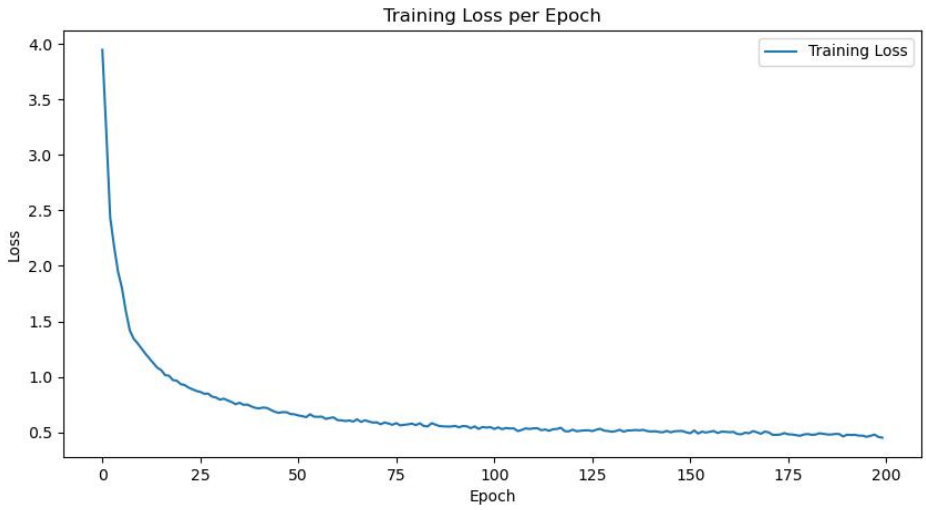
4. 实验结果分析

4. 1 实验结果

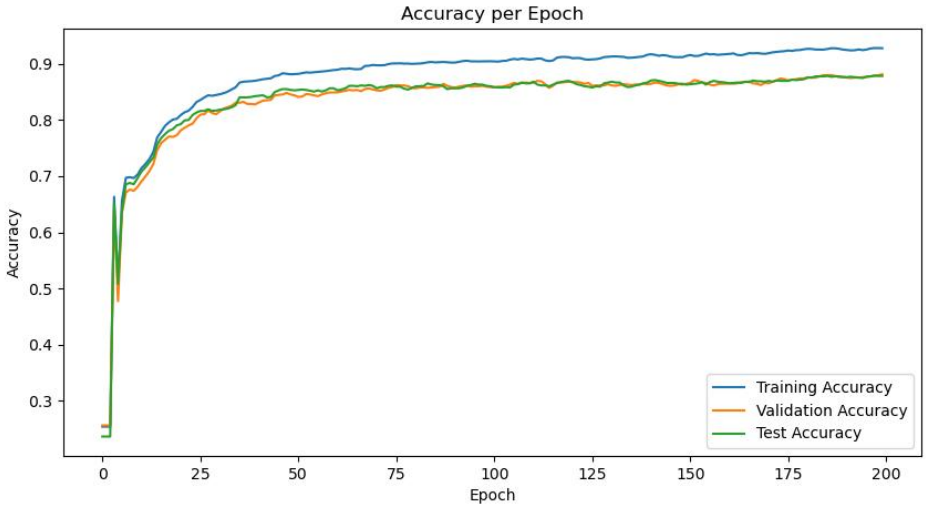
（1）学习率为 0.1，迭代次数为 200 次，R52 数据集

GCN：图卷积神经网络

1. 损失函数曲线：



2. 准确率曲线：

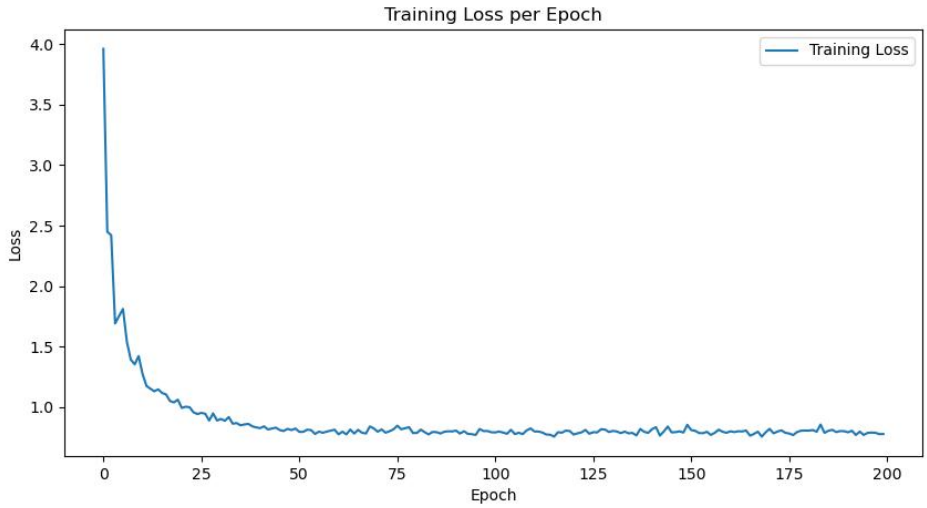


3. 性能指标：

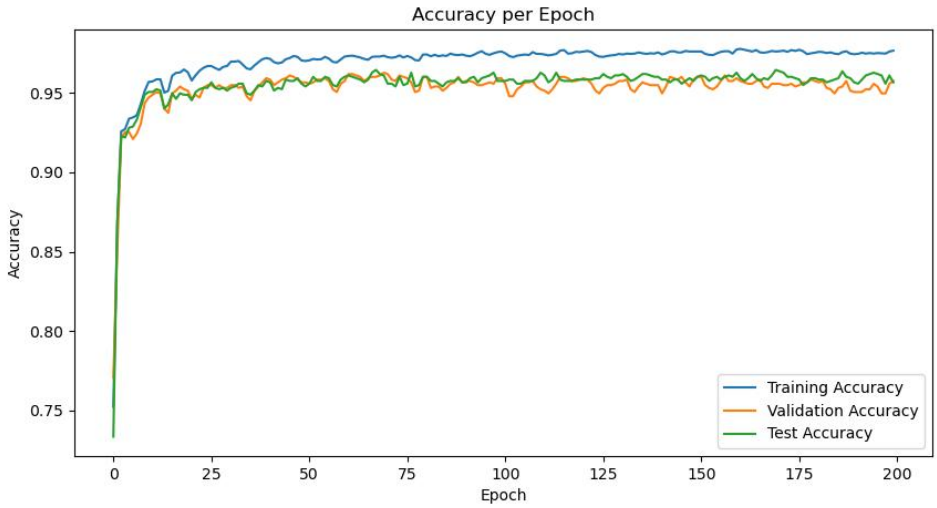
Precision	Recall	F1 Score
0.4172	0.3738	0.3789

GAT：图注意力神经网络

1. 损失函数曲线：



2. 准确率曲线：



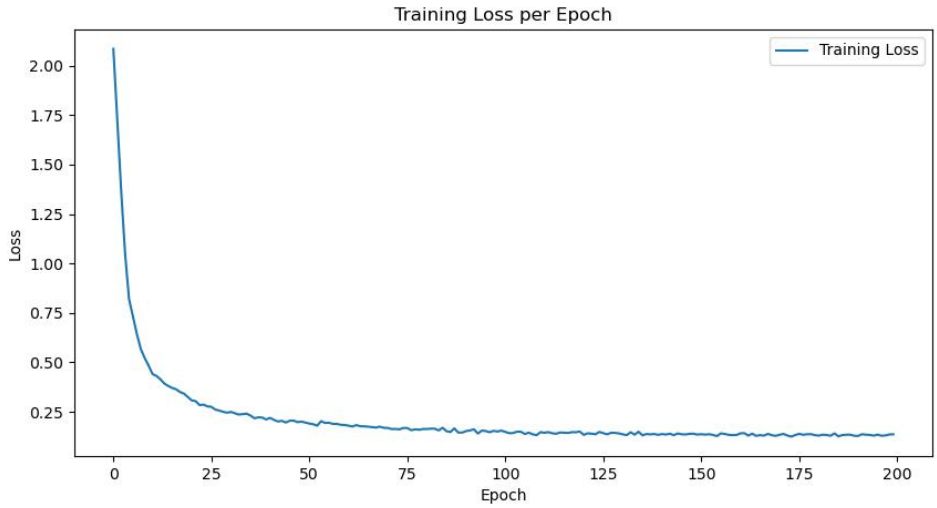
3. 性能指标：

Precision	Recall	F1 Score
0.7210	0.5963	0.6197

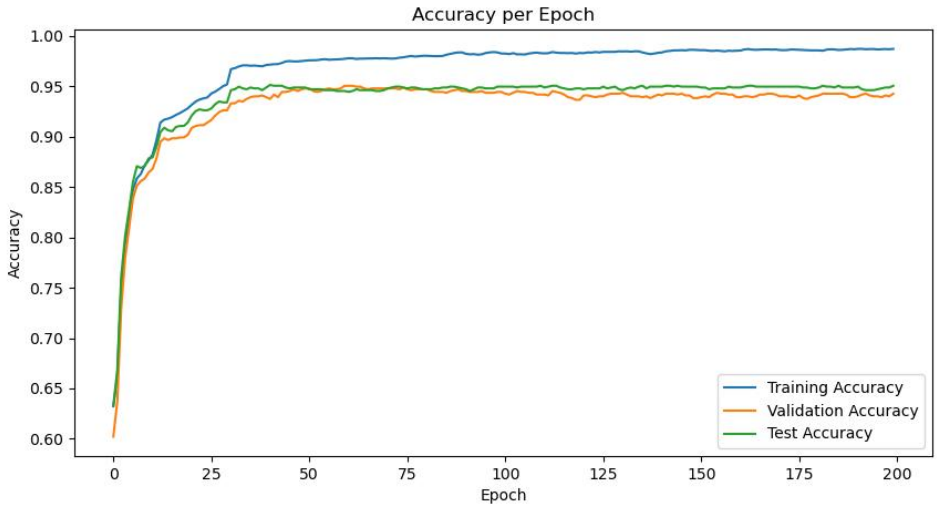
（2）学习率为 0.1，迭代次数为 200 次，R8 数据集

GCN：图卷积神经网络

1. 损失函数曲线：



2. 准确率曲线：

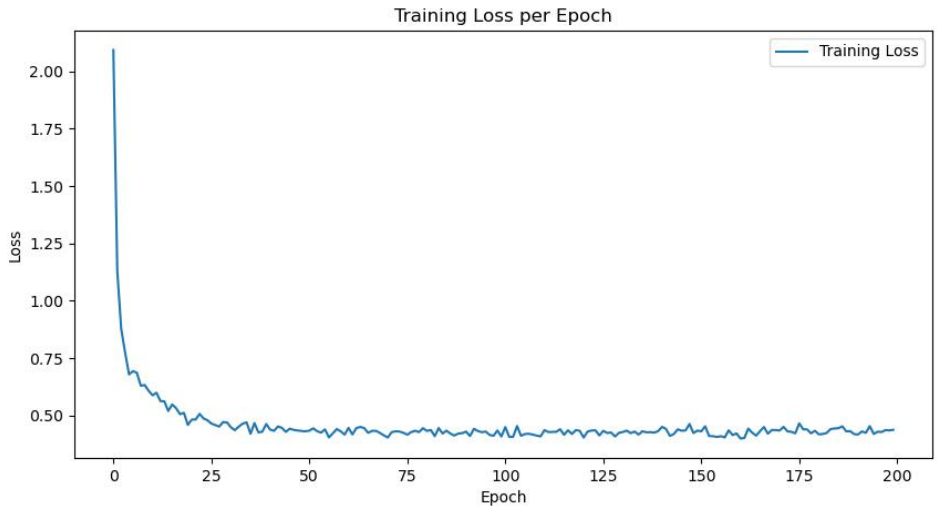


3. 性能指标：

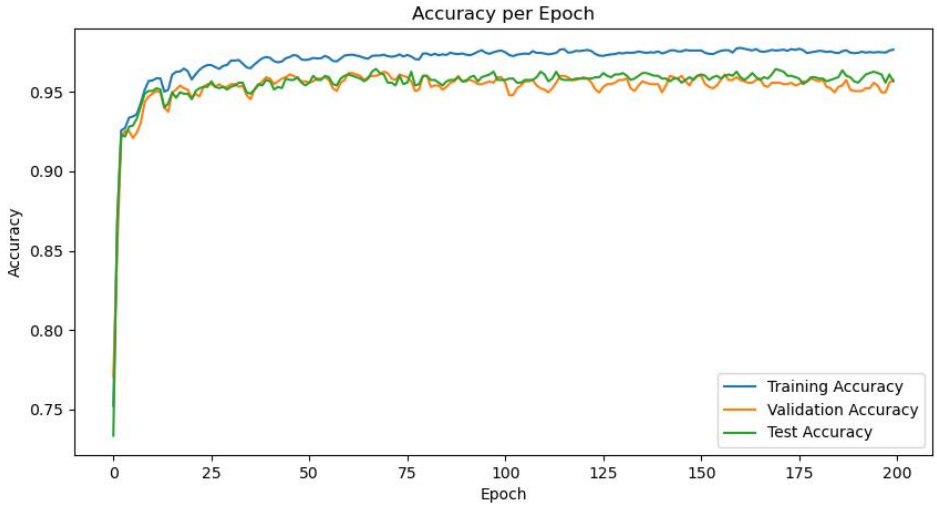
Precision	Recall	F1 Score
0.9528	0.9140,	0.9305

GAT：图注意力神经网络

1. 损失函数曲线：



2. 准确率曲线：



3. 性能指标：

Precision	Recall	F1 Score
0.9571	0.9287	0.9414

4.2 结果分析

4.2.1 性能指标分析

R52 数据集分析

GCN：

Precision（精确率）：0.4172 - 表明在预测为正类的样本中，大约 41.72% 是真正的正类。

Recall（召回率）：0.3738 - 表明模型能够找到 37.38% 的真正的正类样本。

F1 Score：0.3789 - 综合考虑了精确率和召回率，表明模型的整体性能一般。

GAT：

Precision：0.7210 - 相比 GCN，GAT 在预测正类的准确性上有显著提高。

Recall：0.5963 - GAT 在查全率方面也优于 GCN，能够识别出更多的真正正类样本。

F1 Score：0.6197 - 综合精确率和召回率的表现，GAT 的整体性能明显优于 GCN。

R8 数据集分析

GCN：

Precision：0.9528 - 在预测为正类的样本中，大约 95.28% 是真正的正类，显示出很高的准确性。

Recall：0.9140 - 能够找到 91.40% 的真正正类样本，表明模型在查全率方面表现良好。

F1 Score：0.9305 - 说明 GCN 在 R8 数据集上表现出色。

GAT：

Precision：0.9571 - 略高于 GCN，表明 GAT 在预测正类的准确性上略有提高。

Recall：0.9287 - 在查全率方面，GAT 也略优于 GCN。

F1 Score：0.9414 - 综合考虑了精确率和召回率，GAT 在 R8 数据集上的整体性能略优于 GCN。

综合分析

在 R52 数据集上，GAT 明显优于 GCN，这可能是因为 GAT 通过注意力机制能够更有效地捕捉复杂的文本特征，特别是在较为复杂的文本分类任务中。

在 R8 数据集上，两种模型的性能都很高，但 GAT 依然略优于 GCN。这表明即使在相对简单的文本分类任务中，GAT 的注意力机制也可能带来性能上的提升。

总体来看，GAT 在两个数据集上的表现均优于 GCN，这可能归因于其在处理节点之间复杂关系时的优势。

4.2.2 损失和准确率曲线分析

（1）损失曲线分析：

R52 数据集上的 GCN 损失曲线分析

快速下降：损失曲线显示在最初的几个 Epoch 中损失迅速下降，这表明模型快速学习到了数据的特征。

平稳收敛：随着 Epoch 增加，损失逐渐趋于平稳，说明模型在训练中逐渐收敛。

可能的提升空间：由于损失曲线在后期仍略有下降，可能通过增加训练次数或调整学习率等参数来进一步降低损失。

R52 数据集上的 GAT 损失曲线分析

初始下降更快：与 GCN 相比，GAT 的损失曲线在初始阶段下降得更快，这可能表明 GAT

模型对于数据特征的捕捉能力更强。

更低的稳态损失：GAT 在训练过程中达到了更低的稳态损失，这通常是性能更好的迹象。

早期停止可能性：损失在大约 100 个 Epoch 后基本稳定，可以考虑早期停止以避免不必要的计算。

R8 数据集上的 GCN 损失曲线分析

快速收敛：损失曲线下降迅速且在前 50 个 Epoch 内趋于稳定，这表明模型很快适应了 R8 数据集。

损失水平较低：与 R52 数据集上的损失相比，R8 上的损失明显更低，这可能由于 R8 数据集相对简单或者模型与数据集的兼容性更好。

R8 数据集上的 GAT 损失曲线分析

迅速且稳定的收敛：GAT 模型在 R8 数据集上的损失曲线下降非常快，几乎在开始时就迅速达到了稳定状态。

极低的稳态损失：损失曲线显示 GAT 在 R8 数据集上的表现非常好，损失值非常低，表明模型的性能可能非常出色。

综合分析

数据集复杂度：在较为复杂的 R52 数据集上，GAT 的性能优于 GCN，这可能是由于 GAT 的注意力机制在处理更复杂的数据结构时更加有效。

模型适应性：在较为简单的 R8 数据集上，GCN 和 GAT 都表现出了快速收敛的特征，但 GAT 仍然略占上风，显示出较低的损失值。

（2）准确率曲线分析：

GCN 准确率曲线分析

初期准确率提升：GCN 模型的训练准确率在最初的几个 Epoch 中迅速提升，这表明模型能够快速从数据中学习。

准确率稳定性：在经历初始的快速上升之后，准确率趋于稳定，这意味着模型已经从数据中学习到了稳定的特征集合，并在训练集上表现良好。

过拟合的风险：如果训练准确率和验证/测试准确率之间存在较大差距，这可能是过拟合的一个信号。如果训练准确率明显高于验证和测试准确率，则需要考虑增加正则化或使用更多的数据。

R52 数据集上的 GAT 准确率曲线分析

快速且高效的学习：GAT 模型的准确率曲线显示出在训练初期就达到了较高的准确率，并且维持了较小的差距，这表明模型具有较强的学习能力和泛化性能。

更好的泛化能力：如果验证和测试准确率曲线与训练准确率曲线非常接近，这表明 GAT 模型在未见数据上有很好的预测能力。

稳定性：GAT 模型的准确率在训练过程中显示出较好的稳定性，准确率在训练后期变化不大，表明模型收敛良好。

综合分析：

GAT 与 GCN 的比较：在 R52 数据集上，GAT 似乎比 GCN 表现得更好，具体体现在准确率更高和收敛更快。注意力机制可能帮助 GAT 在捕捉文本特征方面更加高效。

模型选择：对于复杂的数据集如 R52，GAT 可能是更优的选择，因为其注意力机制能够更好地处理复杂关系和特征。

4.3 结果解释与理论联系

4.3.1 模型特性分析

GAT：

理论优势：GAT 通过其注意力机制能够为每个节点分配不同的重要性，这使得它在处 理图结构数据时更为灵活。在文本分类中，这意味着 GAT 能够区分哪些单词或短语对分类决策更为关键。

适用情景：在复杂的数据集，如 R52，这种能力尤为重要，因为这些数据集可能包含 更多的噪声和复杂的特征模式。GAT 可以通过为关键的文本特征分配更高的权重，从而提升模型性能。

GCN：

理论基础：GCN 假设图中的节点是同质的，并且通过在节点的邻域内均匀聚合信息来 更新节点的表示。这种方法对于捕捉局部结构信息很有效，但在区分不同节点的重要性方面存在局限。

适用情景：在结构相对简单且关系更直接的数据集，如 R8，GCN 可能已足够捕捉分类所需的关键信息，因此表现良好。

4.3.2 综合理论联系

注意力机制的重要性：在处理文本数据时，特别是在复杂的数据集中，注意力机制允许模型专注于最重要的部分，这与人类在阅读和理解文本时的处理方式类似。

同质性假设的限制：GCN 的同质性假设在简单的数据集上效果可能不错，但在复杂或不规则的图结构中，这种假设可能限制了模型的性能。

于包含复杂关系和需要更细粒度区分的文本分类任务，采用注意力机制的模型可能更为合适。相反，在结构简单、关系直接的数据集上，更传统的 GCN 已足够有效，且更易于计算。

5. 附加题：CNN 和 LSTM 实现文本分类

实验设置部分和 GCN,GAT 区别不大，部分主要介绍以下 CNN 和 LSTM 的模型原理及最终实验结果与 GCN,GAT 的不同。

5.1 模型原理

5.1.1 CNN 实现文本分类

嵌入层

词嵌入：nn.Embedding 层用于将每个单词转换为固定维度的向量。这些向量能够捕捉语义信息，并且在训练过程中学习到单词之间的关系。

卷积层

特征提取：使用多个不同大小的卷积核（由 filter_sizes 参数定义），模型可以提取不同长度的词汇组合的特征。这相当于在文本中应用不同大小的滑动窗口。

多尺度特征：每种大小的卷积核提取不同层次的语义信息，从而能够捕捉到短语或句子的不同方面。

池化层

下采样：max_pool1d 是一维的最大池化操作，它降低了特征的维度并保留了最显著的信号，增加了模型对位置的不变性。

全连接层

分类：最后，通过 nn.Linear 全连接层将卷积层提取的特征映射到分类标签上。

Dropout

正则化：在全连接层前使用 nn.Dropout 以减少模型过拟合的风险。

```
class TextCNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, num_classes, filter_sizes, num_filters):
        super(TextCNN, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.convs = nn.ModuleList(
            [nn.Conv2d(1, num_filters, (k, embedding_dim)) for k in filter_sizes]
        )
        self.dropout = nn.Dropout(0.5)
        self.fc = nn.Linear(len(filter_sizes) * num_filters, num_classes)

    def forward(self, x):
        x = self.embedding(x)  # [N, L, D]
        x = x.unsqueeze(1)  # [N, 1, L, D]
        x = [F.relu(conv(x)).squeeze(3) for conv in self.convs]  # [(N, Co, L), ...] * len(Ks)
        x = [F.max_pool1d(i, i.size(2)).squeeze(2) for i in x]  # [(N, Co), ...] * len(Ks)
        x = torch.cat(x, 1)
        x = self.dropout(x)
        logit = self.fc(x)
        return logit
def tokenize_and_pad_texts(texts, max_len=500):
    # 创建并训练分词器
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(texts)
    # 将文本转换为整数序列
    sequences = tokenizer.texts_to_sequences(texts)
    # 填充序列以获得统一的长度
    padded_sequences = pad_sequences(sequences, maxlen=max_len, padding='post')
    return padded_sequences, tokenizer.word_index
```

5.1.2 LSTM 实现文本分类

嵌入层

词嵌入：nn.Embedding 层将词汇表中的每个单词转换成固定维度的嵌入向量。这些嵌入向量可以捕捉单词之间的语义关系，并在模型训练过程中进一步优化。

LSTM 层

顺序数据处理：LSTM 是一种特殊类型的循环神经网络（RNN），非常适合处理顺序数据，如文本。它能够在序列的不同时间点保存信息，并处理长距离依赖问题。

记忆细胞：LSTM 通过门控机制（包括输入门、遗忘门和输出门）控制信息的流动，允

许网络学习何时忘记旧信息以及何时更新新信息。

隐藏状态：每个时间步的隐藏状态（h_n）捕捉了到目前为止的序列信息。在文本分类中，通常使用最后一个时间步的隐藏状态作为序列的表示。

全连接层

分类决策：nn.Linear 层将 LSTM 的最后一个隐藏状态映射到类别标签上，输出每个类别的得分。

输出

概率分布：F.log_softmax 函数将线性层的输出转换为对数概率分布，表示模型对每个类别的置信度。

文本分类中的作用

捕捉上下文：LSTM 由于其循环结构，特别擅长捕捉文本中的上下文信息，这对于理解句子的整体意义至关重要。

处理不同长度文本：LSTM 能够处理不同长度的输入序列，这在文本数据中非常常见。

长距离依赖：LSTM 特别适用于理解具有长距离依赖关系的文本，如主题较为复杂或结构较为宽松的句子。

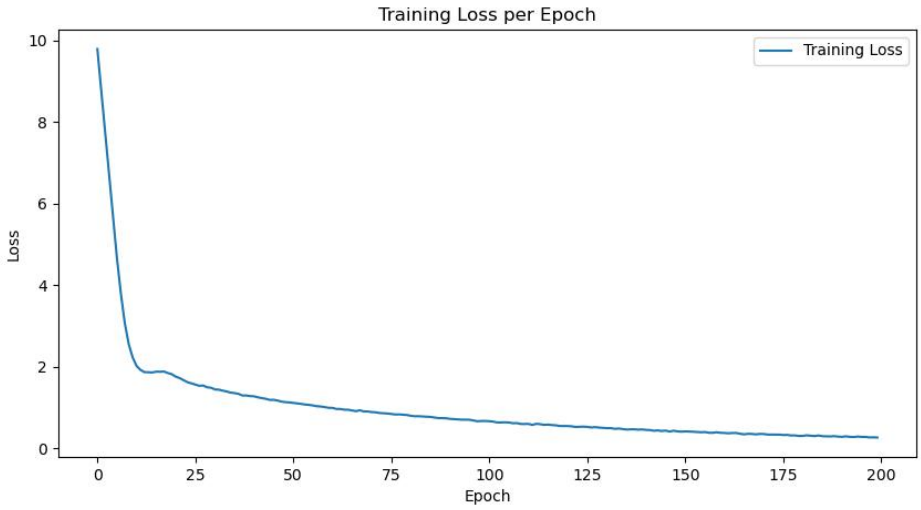
```
class TextLSTM(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, num_classes):
        super(TextLSTM, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers=1, batch_first=True)
        self.fc = nn.Linear(hidden_dim, num_classes)

    def forward(self, x):
        x = self.embedding(x)  # [N, L, D]
        x, (h_n, c_n) = self.lstm(x)  # h_n is the hidden state
        x = h_n[-1, :, :]  # Take the last layer hidden state
        x = self.fc(x)
        return F.log_softmax(x, dim=1)
```

5.2 结果分析

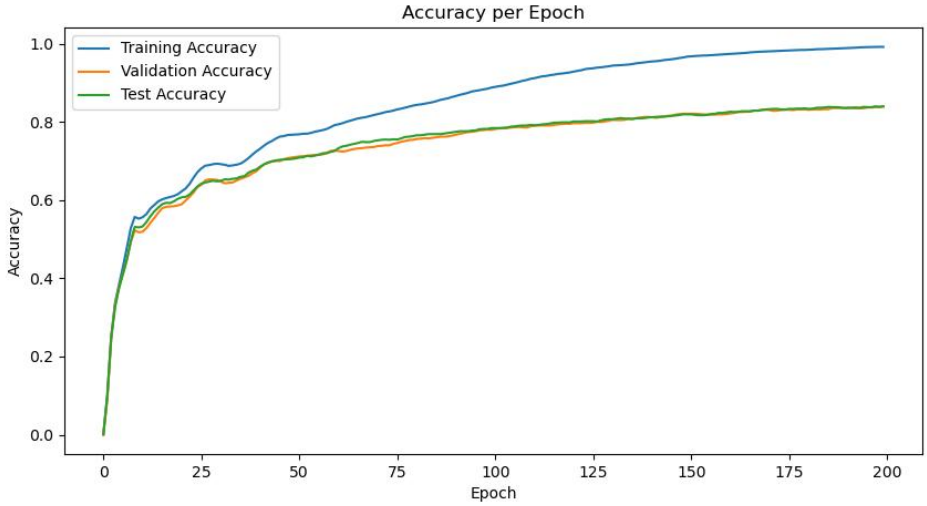
5.2.1 实验结果

（1）CNN，R52 数据集，学习率 0.001，迭代 200 次

1. 损失函数曲线



2. 准确率曲线

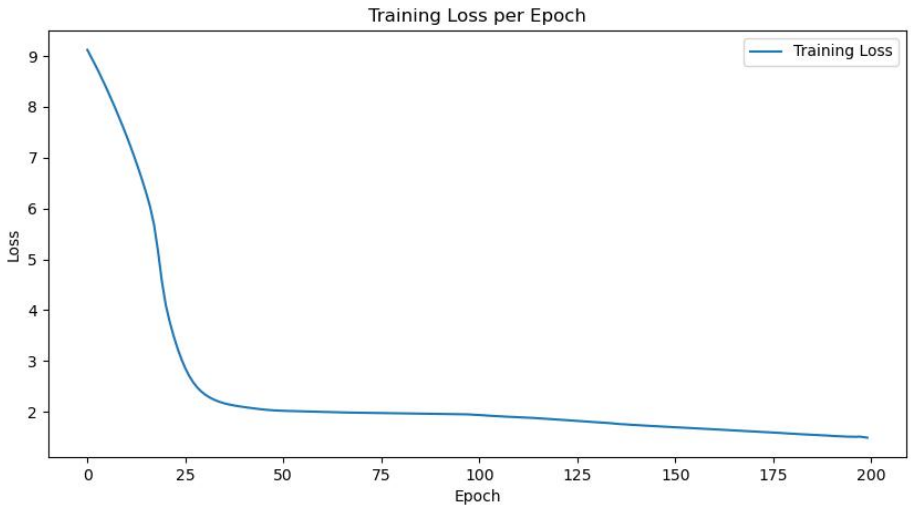


3. 性能指标

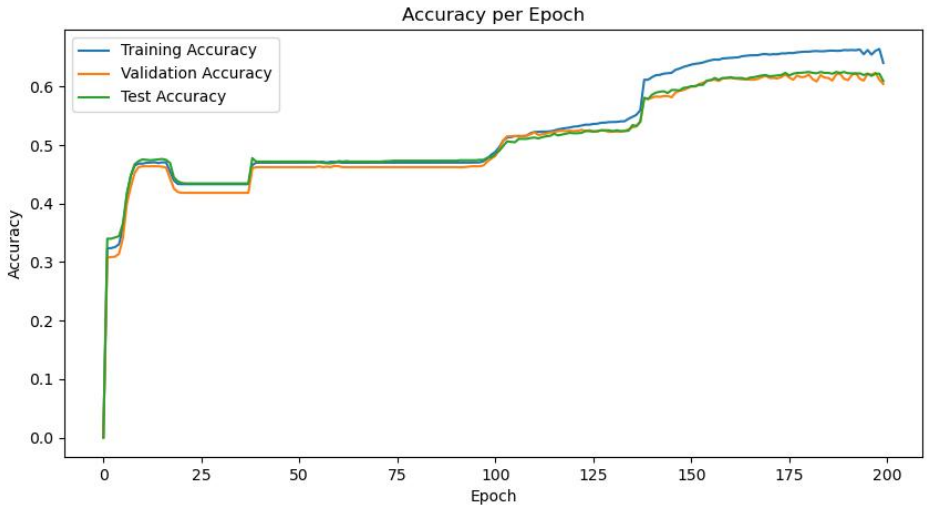
Precision	Recall	F1 Score
0.9903	0.8012	0.8102

（2）LSTM，R52 数据集，学习率 0.001，迭代 200 次

1. 损失函数曲线



2. 准确率曲线

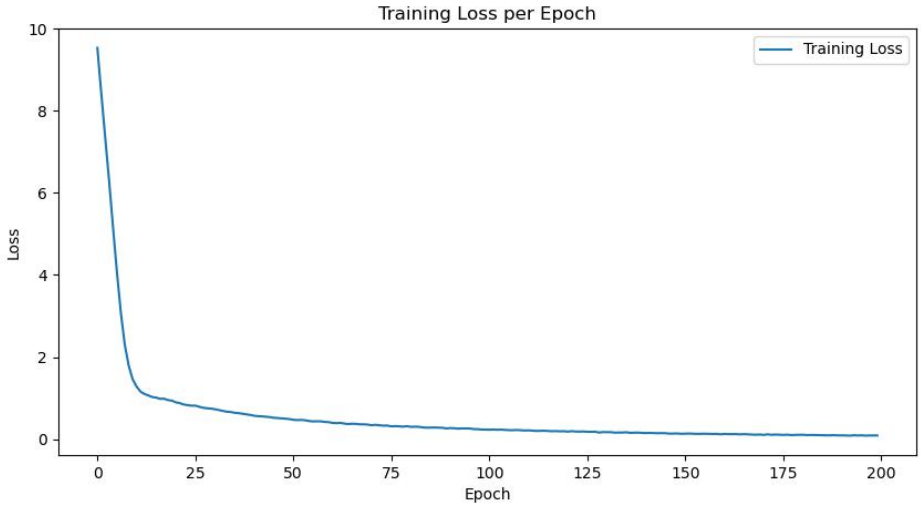


3. 性能指标

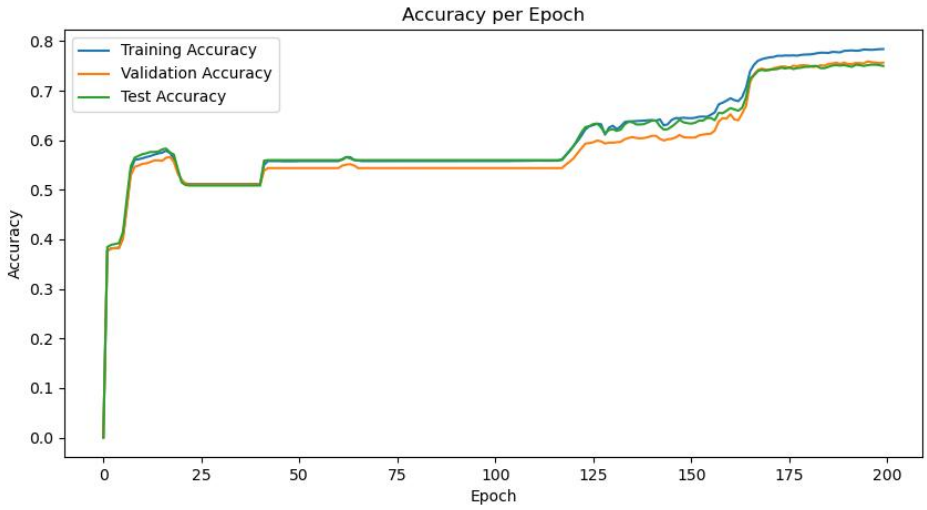
Precision	Recall	F1 Score
0.0248	0.0349	0.0283

（3）CNN，R8 数据集，学习率 0.001，迭代 200 次

1. 损失函数曲线



2. 准确率曲线

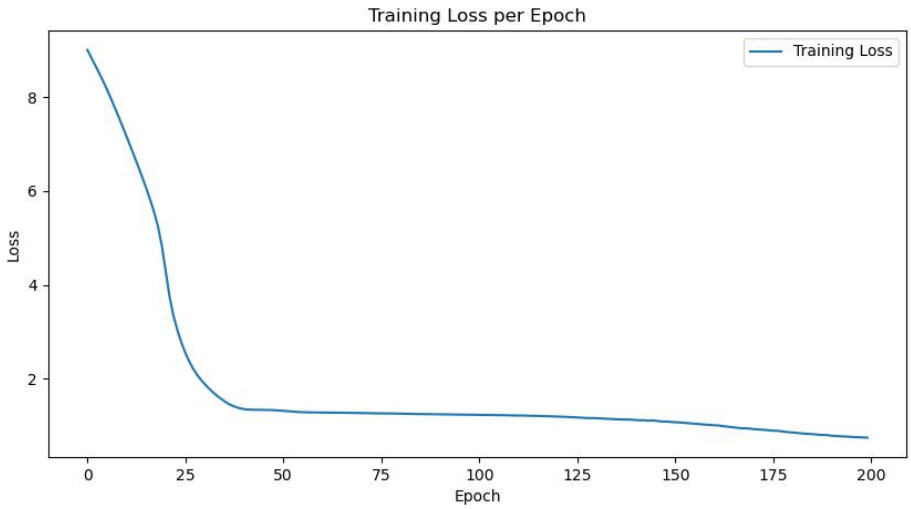


3. 性能指标

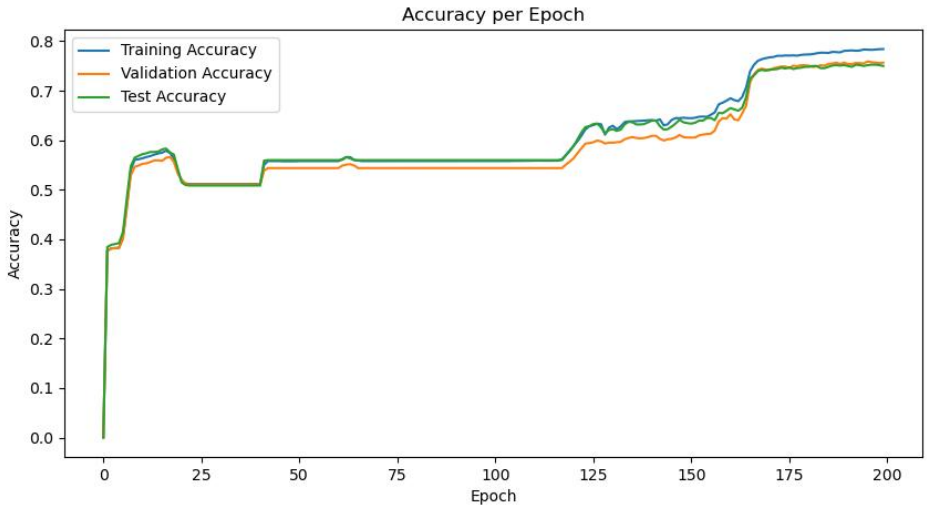
Precision	Recall	F1 Score
0.9675	0.9192	0.9410

（4）LSTM，R8 数据集，学习率 0.001，迭代 200 次

1. 损失函数曲线



2. 准确率曲线



3. 性能指标

Precision	Recall	F1 Score
0.1938	0.2398	0.2106

5. 2. 2 分析与图神经网络差异

1. 计算效率：

CNN 和 LSTM 在进行训练和推理需要的时长远大于 GCN 和 GAT，且需要较大的运算内存。受限于本地机器内存，本实验只能以较小的序列长度进行训练测试，无法完全发挥出 CNN 和 LSTM 的能力。

CNN 和 LSTM 在训练和推理过程中需要较长的时间，特别是对于较长的序列。

GCN 和 GAT 可以通过并行化和图的局部性质来提高计算效率，适用于大型图。

2. 特征提取：

综合准确率来说两种图结构的神经网络性能要优于 CNN 和 LSTM，这和它们进行的特征提取方式有关

CNN 通过卷积核和池化层从局部特征中提取信息。在文本分类中，CNN 可以捕获词汇的局部模式，如 n-grams。

LSTM 通过循环神经网络的隐藏状态从整个序列中提取信息，捕获了单词之间的上下文关系和长距离依赖。

GCN 和 GAT 通过在图上进行消息传递来捕获节点之间的依赖关系。它们依赖于图结构，可以用于捕获节点之间的关联和传播信息。

3. 适用场景

CNN 适用于捕获文本中的局部特征，如情感词汇或短语。

LSTM 适用于处理具有长距离依赖关系的文本，如自然语言生成或情感分析，故对于 R8 和 R52 数据集处理起来效果较差。

GCN 和 GAT 适用于具有复杂关联结构的数据，如社交网络分析或知识图谱。

6. 附加题：异构神经网络分类

题目：思考如何利用异构图神经网络对包含有文本、图像、元数据信息（作者、发表期刊会议等）的多模态异构文档进行分类，通过调研目前存在的方法和策略，设计相应的解决方法，并分析方法的可行性以及创新性。（注：本题目不需要代码实现，要求有新意，并撰写报告。）

6.1 解决方案概述

1. 多模态数据处理：

文本数据：采用自然语言处理技术进行预处理，如使用 BERT 或 Transformer 模型提取文本特征。

图像数据：通过卷积神经网络（如 ResNet）提取图像特征。

元数据：转换为结构化数据形式，可通过嵌入技术转换为数值向量。

2. 构建异构图：

节点：文档、作者、期刊等作为不同类型的节点。

边：表示不同实体间的关系，如文档与作者的归属关系、文档间的引用关系等。

3. 图神经网络模型设计：

根据异构图特点选择合适的 GNN 架构，如异构图注意力网络（HAN）。

设计模型以处理多模态信息，将文本、图像和元数据的特征融合。

4. 分类任务实现：

定义多类别分类任务。

使用 GNN 模型输出进行分类。

5. 创新性分析：

融合多模态数据：集成文本、图像和元数据。

异构图处理：应用 GNN 处理复杂的异构关系。

6.2 实施步骤

1. 数据预处理：
文本：使用 NLP 工具提取关键信息和语义。
图像：使用 CNN 提取特征。
元数据：标准化处理。
2. 图构建：
将处理后的数据映射为图中的节点和边。
定义不同类型的边来表示不同的关系。
3. GNN 模型训练：
设计一个 GNN 模型，考虑节点和边的类型。
训练模型以学习节点表示。
4. 分类与评估：
使用 GNN 模型输出对文档进行分类。
采用准确率、F1 分数等指标评估模型性能。

参考文章：
Multimodal learning with graphs | Nature Machine Intelligence《Multimodal learning with graphs》提供了多模态图学习的基本概念和方法
Heterogeneous Graph Neural Networks for Extractive Document Summarization - ACL Anthology《Heterogeneous Graph Neural Networks for Extractive Document Summarization》介绍了异构图神经网络在文档摘要提取中的应用
Heterogeneous Graph Neural Networks for Extractive Document Summarization - ACL Anthology《Multimodal representation learning on graphs》详细描述了多模态表示学习在图结构中的应用