论文复现报告

1 论文解读

1.1 概述

1.1.1 基本信息

基本信息	详细内容
标题	Cross-links Matter for Link Prediction: Rethinking the Debiased GNN from a Data Perspective
作者	Zihan Luo, Hong Huang, Jianxun Lian, Xiran Song, Xing Xie, Hai Jin
发表刊物	Thirty-seventh Conference on Neural Information Processing Systems
发表年份	2023
关键词	link , GNN , bias

1.1.2 论文概述

本文解决了基于图神经网络（GNN）的**链接预测中存在的偏差问题**，特别关注节点簇间的偏差，即所谓的交叉链接。研究发现现有的GNN模型在处理同一簇内部链接和交叉链接时存在严重的数据偏差。为了解决这些问题，论文提出了一种双GNN框架，可以有效减少偏差，并在端到端的方式下提升模型效用。**具体方法是生成去偏置的节点嵌入，并将其与原始GNN嵌入融合，通过增强的监督信号和动态训练策略来实现**。实验结果表明，该框架不仅能够减少内部链接和交叉链接之间的偏差，还能显著提高整体预测准确率，并且在与其他先进方法的比较中显示出优越性。

1.2 论文内容

1.2.1 论文背景

近年来，GNN在学习图结构数据的潜在表示方面表现出色，被广泛应用于推荐系统、生物信息学和知识图谱等领域，取得了显著的性能提升。然而，现有的链接预测方法往往忽视节点**敏感属性**（如性别或地域）可能带来的偏差，这可能导致信息封闭现象的产生。例如，在社交推荐中，模型倾向于推荐同一地区的用户，限制了用户与外界的连接机会。文章给了一个很直观的例子：在政治新闻推荐中，推荐系统偏好于传递与用户政治信念相符的内容，过滤掉不同观点，从而限制了用户视野。

为了解决这些问题，已经有研究者提出了一些解决方案，如下表：

方法	描述
FLIP	利用对抗学习，鼓励模型预测与敏感属性无关的结果，旨在减少链接预测中的偏见。
CFC	应用组合对抗约束来消除节点嵌入中的偏见，专注于减少与敏感属性相关的偏见。
FairAdj	学习边权重以生成公平的邻接矩阵，确保网络连接表示中的公平性。
UGE	探索从未观察到的图中获取无偏节点嵌入的方法，使用重新加权和正则化方法，减少节点表示中的偏见。

文章也指出现有方法的主要问题：

- 通常修改目标函数以减少偏差，这可能影响模型优化轨迹并导致次优收敛状态。
- 仅关注节点的敏感属性，忽视了基于图拓扑结构的偏差。

1.2.2 解决的问题

GNN在邻域消息聚合中的依赖性导致其更倾向于预测同一社区内的链接（内部链接），而忽视连接不同社区的链接（交叉链接）。研究者统计了三个真实世界数据集中内部链接和跨链接的比例，发现内部链接数量远远超过跨链接。**这种数据偏见可能会因为GNN聚合操作的堆叠而进一步放大，并最终导致链接预测的偏见。**现有的解决方法大多从目标函数入手，可能导致模型次优收敛状态，同时也忽略了图拓扑结构的偏差。

为了解决这一问题，论文提出了一个简单而有效的双结构gnn增强框架。该框架包括两个独立的GNN模型，旨在减少内部链接和跨链接之间的偏见。

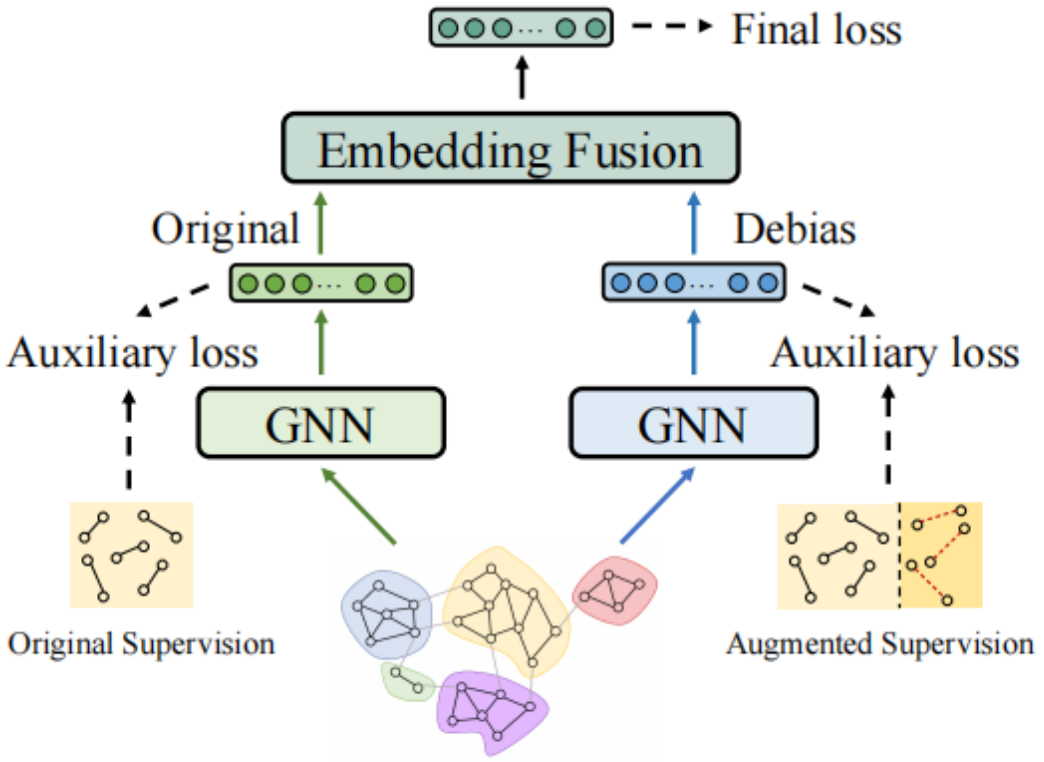


Figure 3: The overview of our framework

本方法有如下流程：

- 研究者首先将整个图划分为**多个具有相似拓扑位置**（连接方式、邻居节点的分布相似）的社区，并将链接区分为内部链接和跨链接。
- 随后，为了减轻内部链接和跨链接之间的数据偏见，论文提出了基于多种规则的监督增强方法，增加了**跨链接的监督信号**，有助于GNN更好地捕捉跨链接的模式并生成无偏节点嵌入。
- 为避免降低效用，论文设计了一个**嵌入融合模块**，采用动态训练策略将原始节点嵌入和无偏节点嵌入进行融合。这样，嵌入融合模块可以有效地保持内部链接的性能，同时减少内部链接和跨链接之间的偏见。

1.2.3 论文方法

本文提出了一种双结构的图神经网络增强框架，旨在减少内部链接和跨链接之间的偏见，同时不损害整体性能。该框架如上图所示，其关键组成部分包括**监督增强**、**双GNN模型**和**嵌入融合模块**。

1.2.3.1 监督增强

监督增强模块通过生成**高置信度的伪跨链接**，增加跨链接的监督信号，帮助GNN模型更好地捕捉跨链接的模式，从而生成去偏置的节点嵌入。这种方法有效地缓解了内部链接和跨链接之间的数据偏差问题。实际上就是在计算损失的时候增加了跨社区的链接数量。文章介绍了两种增强方法。

- 基于Jaccard系数的增强：计算跨社区点对的Jaccard系数，选择最大的K对。$K = k \cdot (|E_{in}| - |E_{cr}|)$
- 基于随机游走的增强：Jaccard可能造成“邻居爆炸”，也很容易只覆盖社区边界的点。所以在原始图上进行随机游走，并记录出现在相同路径中的节点对$\langle \hat{u}, \hat{v} \rangle$。筛选出K个具有**最高出现频率**的跨社区节点对作为增强的监督信号。

在理解监督增强模块时主要有以下几个重点：

- 仅生成伪监督信号，**原始图结构保持不变**，用于GNN中的消息传递。
- 通过增加监督信号，减少内部链接和跨链接之间的偏差。
- 最终的增强监督信号集 E^A是原始监督信号集 E^O 和新构造的边集 E^P的并集。

1.2.3.2 双GNN模型

双GNN架构（Twins GNNs）受到检索式学习范式的启发，其核心思想是提供可以作为模型参考的支持示例。具体而言，通过生成去偏置的节点嵌入，作为框架中减少内部链接和跨链接之间偏见的参考。

使用两个具有相同架构的GNN模型分别对原始监督信号 $\mathcal{E}^O$ 和增强监督信号$\mathcal{E}^A$进行训练。双GNN模型分别生成两种节点嵌入：原始节点嵌入 $\mathbf{Z}^O$和去偏置节点嵌入$\mathbf{Z}^A$。为了确保双GNN模型能够精确生成原始嵌入和去偏置嵌入，采用经典的推荐系统损失函数——BPRLoss（贝叶斯个性化排序损失）来设计辅助损失函数:

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|} \sum_{\langle u,v \rangle \in \mathcal{E}} \log \sigma(r_{u,v} - r_{u,\hat{v}})$$

其中，$\mathcal{E}$可以是 $\mathcal{E}^O$ 或 $\mathcal{E}^A$ 取决于生成的嵌入类型，辅助损失函数分别记为$\mathcal{L}^O$ 和$\mathcal{L}^A$。

σ是激活函数，$\langle u, \hat{v} \rangle$表示负样本，$r_{u,v}$ 表示给定节点对 $\langle u, v \rangle$的预测得分。

注意：

- 在双GNN的嵌入过程中，两个GNN的参数会根据各自的损失函数进行更新。

1.2.3.3 嵌入融合模块

在监督增强阶段设定了几条启发式规则以确保生成的信号具有高置信度，但这一过程不可避免地会**引入噪声**。因此，去偏置节点嵌入$\mathbf{Z}^A$是在不纯净的监督下训练的。如果将 $\mathbf{Z}^A$直接与 $\mathbf{Z}^O$ 进行简单平均融合，尽管跨链接的性能可能会有所提升，但内部链接的性能将显著下降。

所以本文采用了一个嵌入融合模块$F(\cdot)$来融合两种嵌入:

$$\mathbf{H}_i = \text{CONCAT}(\mathbf{Z}_i^O, \mathbf{Z}_i^A)$$

$$\mathbf{Z}_i = F(\mathbf{H}_i, \theta_F) = W_2 \sigma(W_1 \mathbf{H}_i + b_1) + b_2$$

$W_1 \in \mathbb{R}^{D \times 2D}, b_1 \in \mathbb{R}^D, W_2 \in \mathbb{R}^{D \times D}, b_2 \in \mathbb{R}^D$是$F(\cdot)$的可训练参数，记为$\theta_F$以简化表示。$\sigma$是激活函数，$D$是嵌入维度。本文认为在嵌入融合阶段无需部署复杂的网络结构，因此采用了一个简单的**两层MLP**进行融合。

为了优化嵌入融合模块$F(\cdot)$和双GNN模型，本文通过最小化以下目标函数进行训练：

$$\mathcal{L}_F = -\frac{1}{|\mathcal{E}^O|} \sum_{\langle u,v\rangle \in \mathcal{E}^O} \log \sigma(r_{u,v} - r_{u,\hat{v}})$$

本文在嵌入过程中引入了一种动态训练策略，以避免在原始嵌入和去偏置嵌入尚不稳定时对嵌入融合模块进行过度训练。具体来说，在第 t 轮的学习率γ_t^F和训练步长S_t计算公式如下：

$$\gamma_t^F = \alpha * \frac{1}{1+e^{-(t-T)}}$$

$$S_t = \beta * \frac{1}{1+e^{-(t-T)}}$$

其中，α, β 和 T 是控制学习率、训练步长和嵌入融合模块 $F(\cdot)$开始稳定训练的超参数。

1.2.4 作者提到的论文局限

本文提到该方法在以下几个方面存在局限性：

1. **偏差未完全消除**：尽管实验结果表明框架在减少内部链接与跨链接之间的偏差方面取得了显著进展，但偏差并未完全消除。这表明**数据偏差可能不是唯一的原因**，可能还存在其他因素导致偏差。
2. **监督增强的理论基础欠缺**：尽管监督增强在一定程度上起到了正则化作用，提升了内部链接的性能，但**目前缺乏对这一现象的理论基础和严格分析**。作者指出了潜在的正则化效果，但未能提供深入的理论解释。
3. **与反事实学习的潜在联系**：作者发现其工作与反事实学习之间可能存在潜在联系。在其设置中，监督增强引入了大量未观察到的样本，这些样本可能帮助GNN更好地捕捉节点对之间的内在关系，从而提供更准确的预测。然而，关于这一点的讨论还较为初步，**缺乏深入研究和实验证明**。

1.3 实验内容

为了全面展示本文方法的性能，论文作者在三个数据集、六个GNN模型上进行实验，并进行了全面的消融实验以及对比实验。

1.3.1 实验设置

实验在三个真实世界的网络数据集上进行，这些数据集来自SNAP和RecBole的公开数据集。具体选择了两类网络：用户-用户（Epinions, DBLP）和用户-项目（LastFM）。Epinions是一个社交网络，代表用户之间的信任关系；DBLP是一个合作社交网络，代表作者之间的合作关系；LastFM包含用户的听歌记录，代表用户-艺术家的互动。其中，Epinions和DBLP数据集采用基于Jaccard系数的增强方法，而LastFM由于其高密度特点，采用基于随机游走的增强方法。

由于本文提出的方法是一个**与模型无关的框架**，兼容几乎所有基于GNN的链路预测模型，因此选择了六种常见且有效的GNN作为我们的基础模型，包括GraphSAGE、GIN、GAT、LightGCN、PPRGo和UltraGCN。这些模型用于验证框架在减少内部链接与跨链接偏差方面的有效性，同时评估其对整体性能的影响。

实验设置了每个模型在数据集上的**性能指标**：

- **内部链接性能（Internal.）**：在测试集中的内部链接上的预测性能。
- **跨链接性能（Cross.）**：在测试集中的跨链接上的预测性能。
- **总体性能（Overall）**：在整个测试集上的预测性能。

- **偏差（Bias）**：内部链接和跨链接性能的差异，公式为Bias = Internal. − Cross.。

实验通过对比基础模型和应用本文框架后的模型在上述性能指标上的表现，评估框架的效果。

1.3.2 实验任务与结果

在实验部分，本文主要聚焦于四方面的实验：

- 在不同数据集上对比使用本论文框架的模型与不使用框架的模型的效果。
- 使用LightGCN模型进行消融实验。
- 使用SAGE模型与竞争对手进行比较。
- 通过观察去偏后的推荐是否能提供更均衡的推荐，来验证框架在缓解信息茧房方面的潜力。

第一个实验的结果结果显示，该框架在大多数情况下能够提升基础模型在内部链接和跨链接上的性能，并降低内部链接和跨链接之间的偏差。

消融实验探讨了以下几个方面对框架的影响：

1. 监督增强的影响。
2. 嵌入融合模块的影响。
3. 双GNN模型中的辅助损失函数的影响。
4. 动态训练策略的影响。

本文通过移除或替换框架的不同组件，分析了每个组件对模型性能和偏差减轻的贡献。

在第三个实验中，将本文提出的方法与其他几种减少内部链接和跨链接偏差的方法进行了对比，包括FairWalk、CFC、FairAdj、FLIP和UGE。实验结果显示，本文的方法在总体性能和偏差减轻方面均达到或超过了其他方法。

在最后一个实验中，通过观察去偏后的推荐是否能提供更均衡的推荐，来验证框架在缓解信息茧房方面的潜力。结果显示，去偏后的GNN模型推荐列表的内部链接比例更接近历史数据，表明该方法能有效减少推荐中的偏差。通过可视化子图，展示了去偏前后社区之间的连接情况，结果表明去偏后社区间的连接更加紧密，表明该方法能有效缓解信息茧房现象。

1.4 思考

1.4.1 论文优点

本文考虑了数据偏差对链接预测的影响，并提出了的通用的框架且在多个数据集上有良好的性能。

本文的优点具体如下：

1. **有效减轻偏差**：本文提出的方法能够显著减轻内部链接和跨链接之间的偏差，这对改善图神经网络（GNN）在链接预测任务中的公平性和准确性起到了重要作用。
2. **模型无关性**：所提出的方法是模型无关的，能够兼容几乎所有基于GNN的链接预测模型。
3. **无需修改损失函数**：通过监督增强模块，增加了伪跨链接，从而指导生成无偏的节点嵌入，无需修改损失函数，避免了模型优化的次优收敛状态。
4. **可解释性和可视化**：通过图的可视化展示，证明了所生成的嵌入在连接社区方面的有效性，降低了社区之间的隔离现象，增强了结果的可解释性。

1.4.2 论文缺点与改进想法

复现完论文后，除了作者所提到几个缺陷，我认为本篇论文还有如下缺点：

1. **考虑单一**：论文只考虑了数据偏差对结果偏差的影响，但是结果表明并没有完全消除结果偏差。在后续的实验中可以尝试添加**对比模块来引导降低偏差**。

2. **只考虑了结构角度增强**：在增强模块中，作者提出使用Jaccard和随机游走算法进行伪边生成来增加监督信号，但是这两种方法**只是考虑到图的结构相似性**。后续可以**通过自编码器或者图编码器对图结构进行重构，新生成的边作为监督信号**，这种方法会增加训练开销但是可以构建出置信度更高的监督信号。

2 论文复现

2.1 环境配置

为了加载原文作者处理好的图数据，在复现的时候，必须保证环境与作者所提供的严格一致。配置环境的时候，由于`dgl_cu102==0.9.1.post1`和`torch==1.12.1+cu102`无法直接通过conda或者pip安装，只能从官网的历史版本中找到对应的.whl在本地进行安装。在下面的部分中我将完整的展示环境配置过程。

1. 创建虚拟环境：

```
conda create --name cross python=3.7.6
conda activate cross
python --version
```

2. CUDA安装

先检查autodl提供的cunda版本：

```
(cross) root@autodl-container-a899408bbc-33e29197:~/autodl-tmp/Cross-links-Bias# conda search cudatoolkit
Loading channels: done
# Name                       Version           Build  Channel
cudatoolkit                      7.5               0  anaconda/pkgs/free
cudatoolkit                      7.5               2  anaconda/pkgs/free
cudatoolkit                      8.0               1  anaconda/pkgs/free
cudatoolkit                      8.0               3  anaconda/pkgs/free
cudatoolkit                      9.0      h13b8566_0  anaconda/pkgs/main
cudatoolkit                      9.0      h13b8566_0  pkgs/main
cudatoolkit                      9.2               0  anaconda/pkgs/main
cudatoolkit                      9.2               0  pkgs/main
cudatoolkit                 10.0.130               0  anaconda/pkgs/main
cudatoolkit                 10.0.130               0  pkgs/main
cudatoolkit                 10.1.168               0  anaconda/pkgs/main
cudatoolkit                 10.1.168               0  pkgs/main
cudatoolkit                 10.1.243      h6bb024c_0  anaconda/pkgs/main
cudatoolkit                 10.1.243      h6bb024c_0  pkgs/main
cudatoolkit                  10.2.89      hfd86e86_0  anaconda/pkgs/main
cudatoolkit                  10.2.89      hfd86e86_0  pkgs/main
cudatoolkit                  10.2.89      hfd86e86_1  anaconda/pkgs/main
cudatoolkit                  10.2.89      hfd86e86_1  pkgs/main
cudatoolkit                 11.0.221      h6bb024c_0  anaconda/pkgs/main
cudatoolkit                 11.0.221      h6bb024c_0  pkgs/main
cudatoolkit                   11.3.1      h2bc3f7f_2  anaconda/pkgs/main
cudatoolkit                   11.3.1      h2bc3f7f_2  pkgs/main
cudatoolkit                   11.8.0      h6a678d5_0  anaconda/pkgs/main
cudatoolkit                   11.8.0      h6a678d5_0  pkgs/main
```

```
conda install cudatoolkit==10.2.89
```

3. 对应pytorch安装

```
wget https://download.pytorch.org/whl/cu102/torch-1.12.1%2Bcu102-cp37-cp37m-linux_x86_64.whl
pip install torch-1.12.1+cu102-cp37-cp37m-linux_x86_64.whl
```

4. 对应dgl库安装

在conda官网上找到：

dgl-cuda10.2	0.9.1post1	py37_0	dgl-cuda10.2-0.9.1post1-py37_0.tar.bz2

```
wget https://conda.anaconda.org/dglteam/linux-64/dgl-cuda10.2-0.9.1post1-py37_0.tar.bz2
conda install dgl-cuda10.2-0.9.1post1-py37_0.tar.bz2
```

依然安装不成功，只能手动解压：

```
tar -xvjf dgl-cuda10.2-0.9.1post1-py37_0.tar.bz2
cp -r lib/python3.7/site-packages/* /root/miniconda3/envs/myenv/lib/python3.7/site-packages/
```

需要注意的是**算力为8.x的显卡使用的cuda版本应该大于等于11.0**，在本论文的源码中使用的CUDA版本为10.2，所以在本实验中GPU也最好与源码中的保持一致，使用**NVIDIA V100**。

2.2 代码架构

Ablation	对比实验
Comparison	对比实验
config	Initial commit
img	Initial commit
script	对比实验
.gitignore	Initial commit
README.md	Initial commit
ablation.py	Initial commit
augmentation.py	Initial commit
e2e_main.py	对比实验
lightgcn.py	对比实验
model.py	Initial commit
preprocess.py	对比实验
process.py	对比实验
requirements.txt	Initial commit
test.py	Initial commit
utils.py	Initial commit
记录.md	对比实验

在本论文的代码中，`script`文件夹是运行不同模型的sh文件，`augmentation.py`介绍了三种添加伪边的增强算法：计算Jaccard系数、随机游走、随机添加。`model.py`中包含了融合模型以及多种GNN模型。`preprocess.py`包含加载数据、louvain算法、数据集划分的函数。`process.py`包含训练以及测试函数。

2.3 复现数据集结果

由于我只租到了一张A100显卡，后期增加了一张，大多数实验的训练时间超过四小时，所以只在`Epinions`数据集上进行了实验。所以所有的增强算法都是Jaccard，以Hits@50结果的百分比作为最终结果。

在每个模型中的实验结果如下：

Model	Internal	Cross	Overall	Bias
gat	36.83	34.33	35.94	2.50
gin	43.06	40.01	41.97	3.05
sage	34.51	33.66	34.21	0.85
ultragcn	45.79	45.12	45.55	0.67
LightGCN	50.45	46.08	49.25	4.37

我们可以和论文给出的结果进行比较：

Model	Internal	Cross	Overall	Bias
SAGE	36.98	34.45	36.08	1.63
GIN	40.56	39.39	40.20	1.26
GAT	39.58	36.13	38.35	3.45
UltraGCN	44.13	43.66	43.96	0.67
LightGCN	51.49	45.31	49.29	6.18

可以发现，在大多数模型中，我的结果和论文结果有一定的一致性，例如`UltraGCN`的偏差结果和论文的结果一致。具体来说，`SAGE`、`GIN`、`GAT`相较于论文的结果，我的结果相对偏低。`LightGCN`、`UltraGCN`的结果是我的复现结果略高于论文结果。

整体来说对`Epinions`数据集复现的结果与论文给出的结果相近，算是一次成功的复现。

2.4 复现消融实验

按照论文中提到的方法，我对每个模块进行了如下的消融实验复现：

- `监督增强`对论文方法框架的影响：

  将通过Jaccard系数选择的增强的交联信号替换为跨不同社区的随机节点对。设置`true_aug=0`，进行虚假增强，随机选择前augment_size条增强边

- `嵌入融合模块`对论文方法框架的影响：

  以一种简单的方式将ZA融合到ZO中，**简单的平均**。

- 等式中`两个辅助损失函数`的影响：

  注释掉e2e_train的前半部分对双GNN的训练过程。

- 训练嵌入融合模块过程中`动态训练策略`的影响：

  设置一个**固定的学习速率γ和训练步骤S**

```
# LR = args.alpha * 1 / (1 + math.exp(-EPOCH + args.threshold))

# STEP = int(args.beta * 1 / (1 + math.exp(-EPOCH + args.threshold))) + 1

LR = 0.01

STEP = 12
```

最终结果如下：

	Epinions Internal.	Epinions Cross.	Epinions Overall	DBLP Internal.	DBLP Cross.	DBLP Overall
- Augment	48.25	40.99	45.02	92.65	60.68	85.13
- Fusion	47.79	40.45	45.18	85.89	51.36	77.76
- Auxiliary Loss	49.44	43.26	47.24	90.92	59.29	83.47
- Dynamic	45.39	42.03	44.19	74.29	34.18	64.84

这是论文中提到的结果：

	Epinions			DBLP		
	Internal.↑	Cross.↑	Overall↑	Internal.↑	Cross.↑	Overall↑
Original	46.43	37.11	43.11	85.95	47.41	76.88
Ours	**51.49**	45.31	**49.29**	**93.55**	65.33	**86.90**
- Augment	49.43	40.97	46.77	92.76	60.97	85.24
- Fusion	44.31	**46.39**	44.87	72.92	**66.97**	71.53
- Auxiliary Loss	49.63	41.55	46.75	85.31	60.61	84.85
- Dynamic	48.63	41.58	46.12	93.09	59.08	85.08

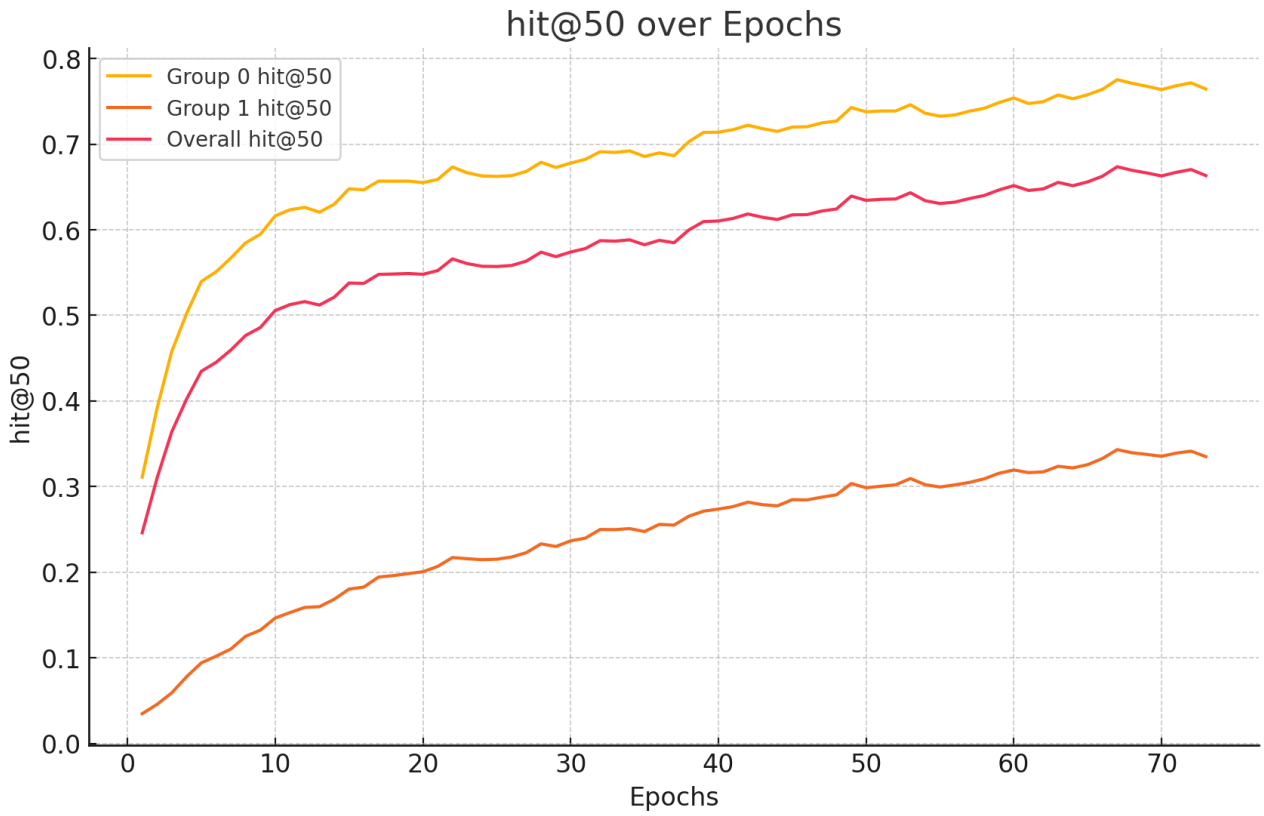
整体来说，我的消融实验结果与论文给出的结果有着一样的趋势。主要的不同在于，我使用平均融合替换融合模块之后没有得到预期的**Cross>Internal**，结果是二者的差值反而更大了，**交叉链接的结果更差了**。这可能是因为平均融合这种方法更容易保留原始嵌入的特征，而增强后的特征表示被削弱，这也印证了Fusion的重要性。

2.5 对比实验

除了对论文中的方法进行复现，我也复现了LightGCN对数据进行推荐的结果：

Dataset	Internal	Cross	Overall
Epinions	45.82	36.72	42.58
DBLP	77.54	34.32	67.37

相较于论文使用LightGCN进行对比的结果：Epinions(46.43,37.11,43.11), BDLP(85.95,47.41,76.88)。Epinions数据集上的结果与我的结果相差不大，**我的结果略低于论文给出的对比结果**。DBLP数据集上，我的LightGCN推荐结果比较差，比论文的结果平均低了10%。这可能是因为没有细调学习率等参数，导致训练**陷入局部最优解而导致早停**。下图展示了在DBLP数据集上的训练效果：



2.6 改进实验

通过观察实验结果发现，论文方法的结果**依然存在着链接预测偏差**，即内部链接的预测依然优于交叉链接的预测。所以我尝试在融合模块中添加一个对比损失，来引导模型关注更多交叉链接。

```
# 对比嵌入
contrastive_loss = torch.nn.functional.mse_loss(embedding, emb_aug.detach())

# 总损失
loss = pos_loss + neg_loss + contrastive_loss
```

我将融合后的特征Z与增强特征Z_A进行对齐，计算了MSE损失后添加到总的损失。

最终的实验结果如下：

Dataset	Internal	Cross	Overall
Epinions	51.29	49.44	50.63
DBLP	88.82	60.05	82.05

与作者给出的结果对比（51.49，45.31，49.29）、（93.55，65.33，86.90）

可以看出，在`Epinions`数据集上，我的改进取得了一定成效，在几乎不降低内部链接性能的情况下，提升了交叉链接的性能，整体性能也有所提高。但是在`DBLP`数据集上，出现了性能下降，我认为主要有两点原因：

1. 由于考核后期我才用到DBLP数据，所以对参数没有进行调整，导致模型没有收敛到最优。
2. 我只是用来固定的相加方式，在DBLP数据集中可能对比损失的数量级较大，所以模型更多关注降低对比损失的方向而忽视了对BPRLoss的优化。后续可以尝试**给对比损失乘一个可学习的参数**，来动态调整对比损失的大小。

3 感悟与说明

3.1 收获

通过阅读这篇论文和复现实验，我了解了消融实验的方法，对推荐系统的之后有进一步增强。同时，作者从数据偏差角度考虑问题也帮我拓宽了思路。如果是在之前，我肯定想着是通过对比或者对抗，引导损失函数去关注交叉链接，相较于作者这种方法有些因小失大了，而且眼界不够宽阔。其实就是应该从本质上想：**造成预测偏差的原因就是本来的数据偏差**！

3.2 提交说明

- improve.py为改进代码，lightgcn.py为对比实验代码，训练日志可以在Log文件夹查看。
- README.md记录了消融实验的方法与结果。
- Github仓库链接：[复现Cross-links-Bias论文]