lab3

实验过程

1. 加载对应数据集：使用dgl.data.citation_graph中的load函数，加载实验用到的数据集，之后分别提取图，特征，边关系，标签等信息，并划分数据集
2. 建立模型：建立GCN模型，使用两层，并手动完成图卷积层。
3. 训练模型，将节点分类与链路预测分别作为两个任务，划分数据集与训练模型
4. 按照TOP1 acc与AUC作为指标，得出结果

关键代码展示

1. 数据集加载

   ```
   g = data[0]
   u, v = g.edges()
   features = torch.FloatTensor(g.ndata['feat'])
   labels = torch.LongTensor(g.ndata['label'])
   mask = torch.BoolTensor(g.ndata['train_mask'])
   return g, u, v, features, labels, mask
   ```

   加载特征，边关系，标签等信息
2. 模型搭建

   ```
   class GCN(nn.Module):
   def __init__(self, in_feats, hid_feats, out_feats, dropedge_prob=0.5):
       super(GCN, self).__init__()
       self.conv1 = GraphConv(in_feats, hid_feats)
       self.conv2 = GraphConv(hid_feats, out_feats)
       self.dropedge_prob = dropedge_prob

   def forward(self, g, features):
       #g,features = self.dropedge(g, features)
       h = F.relu(self.conv1(g, features))
       h = self.conv2(g, h)
       return h
   ```

   该GCN网络有两个图卷积层，之后再手动实现图卷积层

```
class GraphConv(nn.Module):
def __init__(self, in_feats, out_feats):
    super(GraphConv, self).__init__()
    self.linear = nn.Linear(in_feats, out_feats)

def forward(self, g, features):
    g.ndata['h'] = features
    g.update_all(fn.copy_u(u='h', out='m'), fn.sum(msg='m', out='h'))
    h = g.ndata['h']
    # 添加自环
    h = h + 0.2*features
    #pair_norm
    norm = torch.pow(g.in_degrees().float().clamp(min=1), -0.5)
    norm = norm.to(features.device).unsqueeze(1)
    h = h * norm
    return self.linear(h)
```

调参分析

由于cora的表现好于citeseer，因此将citeseer作为模型好坏的指标

1. 自环修改
    i. 不添加自环：
       Validation Accuracy 0.7293
       Valid AUC: 0.8264853826210116
    ii. 添加自环
        a. h = h + h'
           Validation Accuracy 0.7308
           Valid AUC: 0.9209254614838063
           可以看出，添加自环之后，由于归一化拉普拉斯矩阵的最大特征值变小，对于链路预测任务有了更好地提升
        b. 采用梯度的方法

```
global i
    # 添加自环
    if i < 30:
        prob = 1
        prob1 = 0.6
    elif i < 60:
        prob = 0.8
        prob1 = 0.8
    else:
        prob = 0.6
        prob1 = 1
    i += 1
    h = prob1*h + prob*features
```

Validation Accuracy 0.7368

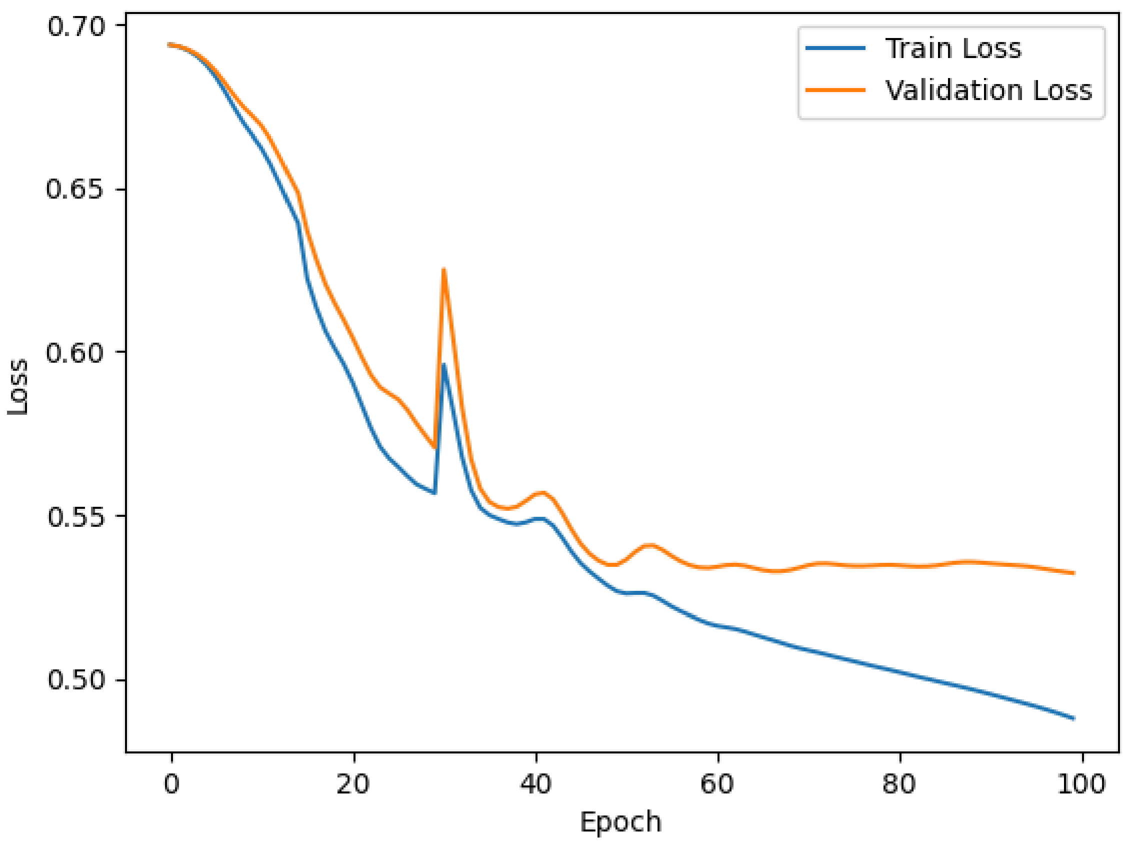
Valid AUC: 0.9063669002122143

2. 网络宽度

   i. 16：

   Validation Accuracy 0.7368

   Valid AUC: 0.8611760720117071



会发现出现较大的震荡，继续增加层数

ii. 32：

Validation Accuracy 0.7233

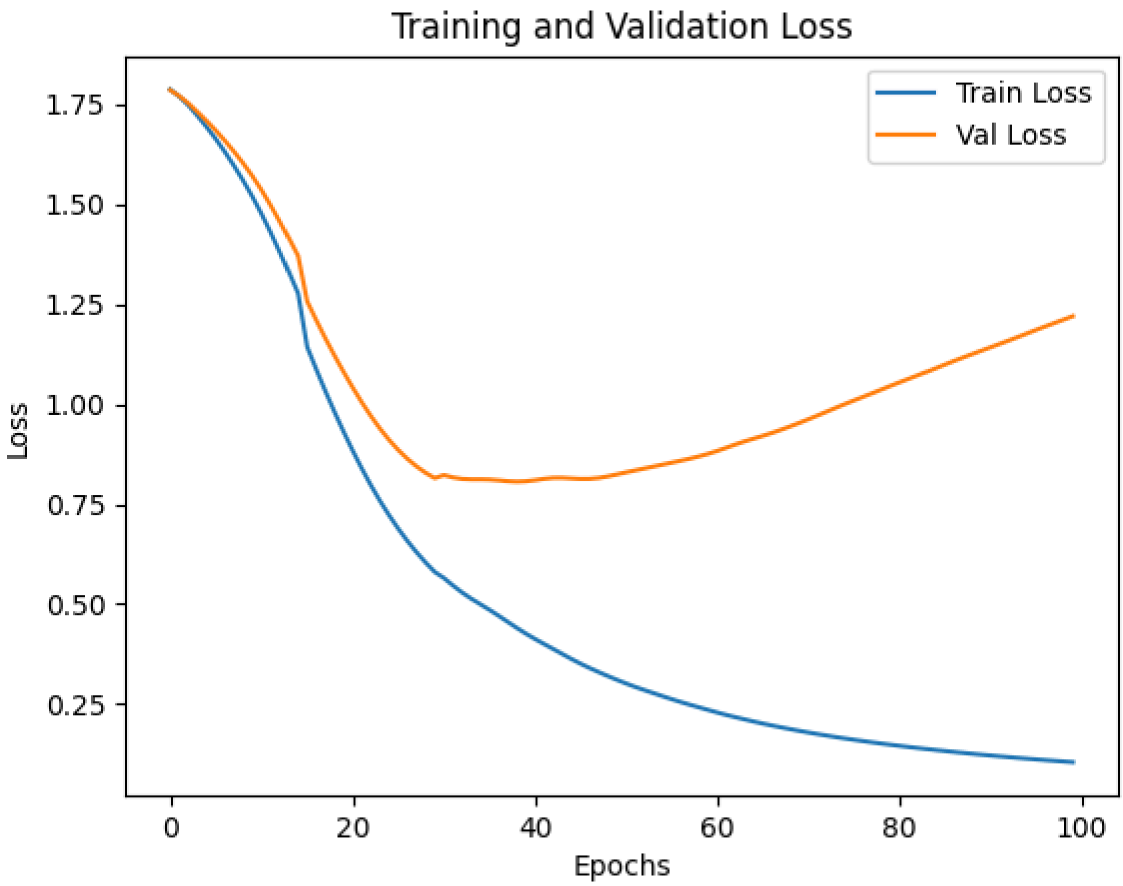
Valid AUC: 0.8968054921631274

iii. 64：

Validation Accuracy 0.7188

Valid AUC: 0.8532145058605973

发现结果下降，之后再观察损失图像



发现过拟合情况更为严重，因此32是一个比较合适的层数

3. 层数

i. 使用2层

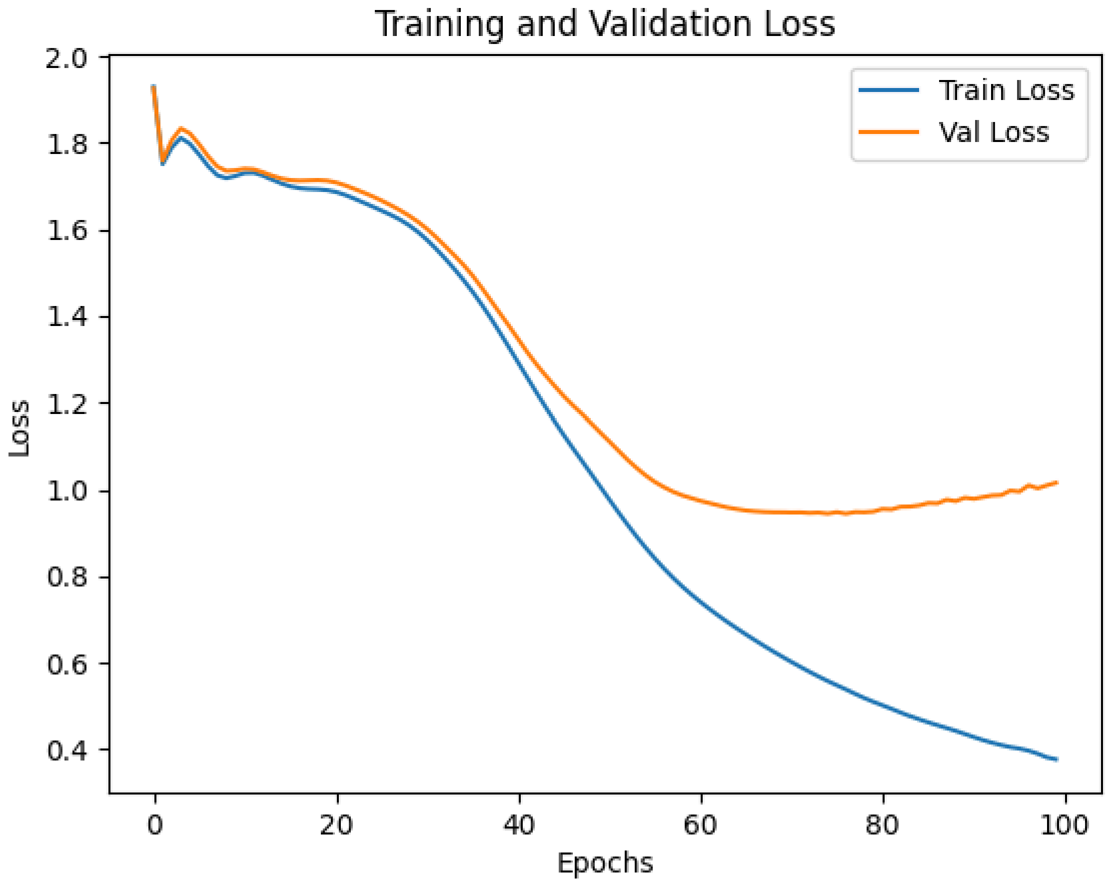
Validation Accuracy 0.7474

Valid AUC: 0.895930284536587

ii. 使用4层

Validation Accuracy 0.7248

Valid AUC: 0.8387312312665572

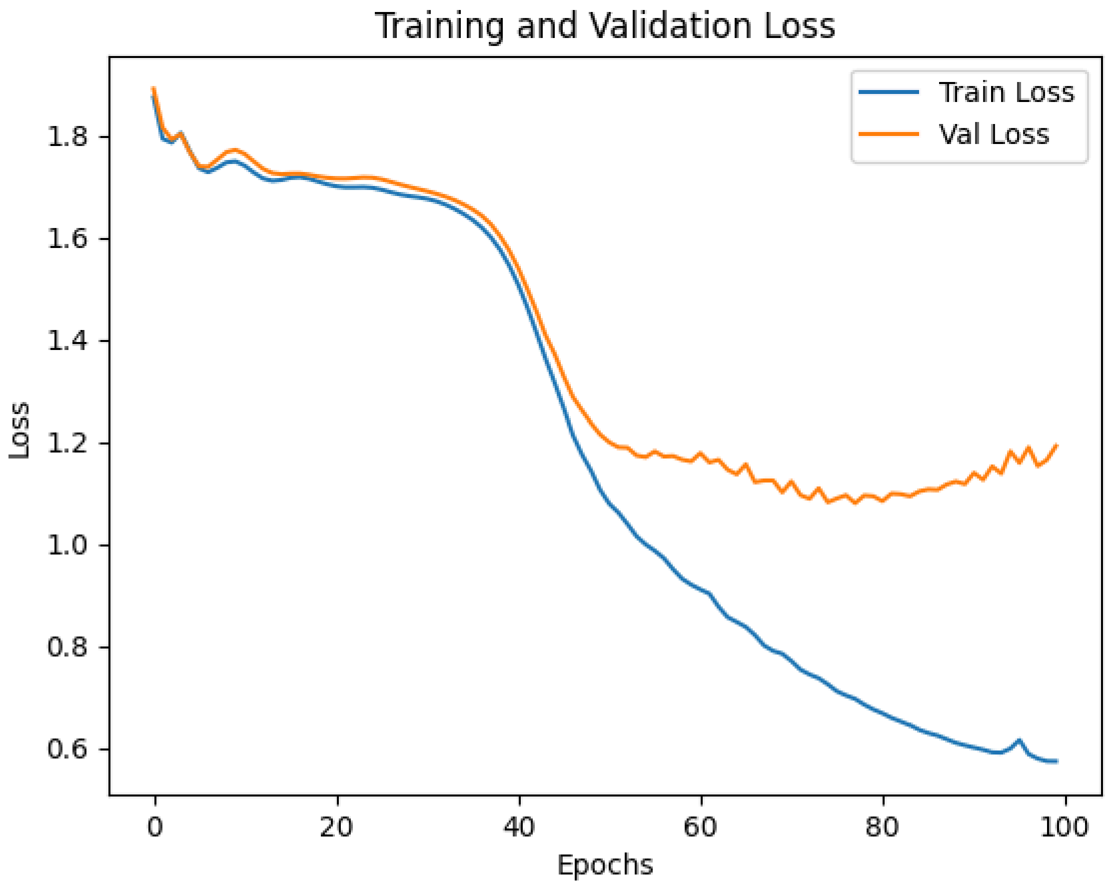


发现过拟合现象更为严重

iii. 使用6层

Validation Accuracy 0.6797

Valid AUC: 0.8447741634944311

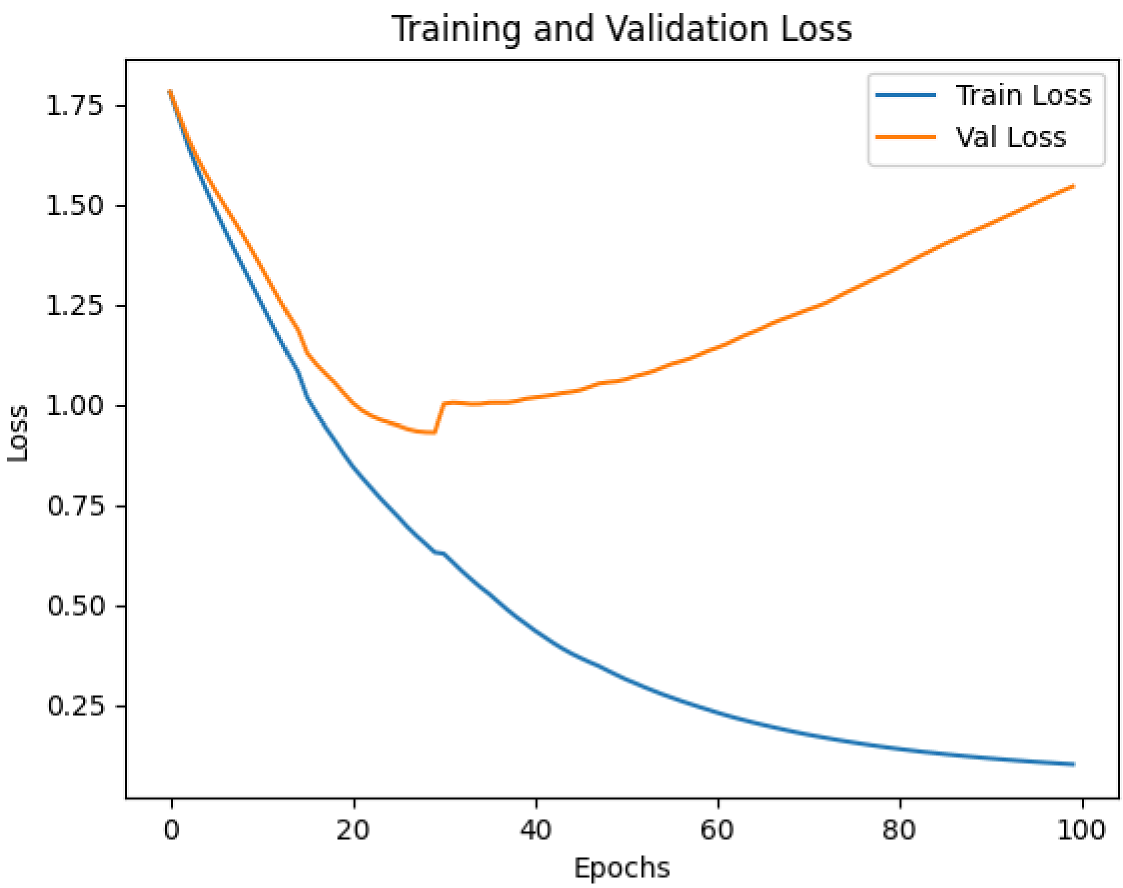


发现模型震荡明显，增加深度会增加梯度在网络中传播的路径长度，这可能导致梯度消失或梯度爆炸的问题。进而导致参数更新变得剧烈

4. pairnorm
    i. 不添加pairnorm

    Validation Accuracy 0.7323

    Valid AUC: 0.922704109240969



发现后面的epoch的过拟合问题较为严重，因此添加pairnorm来解决过拟合问题

ii. 添加pairnorm

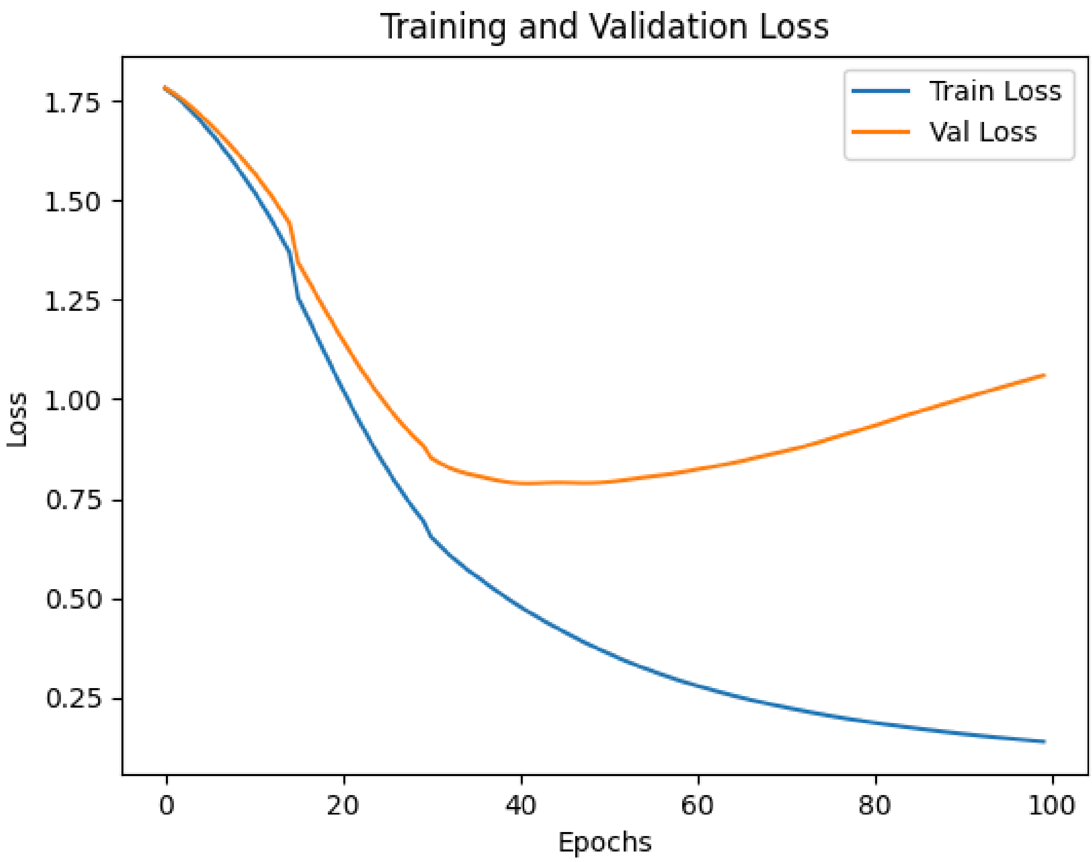
对于归一化的参数调整，主要调整代码中的min值。减小min值会导致更大的入度值，进而得到更小的归一化系数，使得特征矩阵的值被更多地归一化调整，可以增强归一化的效果。

```
norm = torch.pow(g.in_degrees().float().clamp(min=1), -0.5)
norm = norm.to(features.device).unsqueeze(1)
h = h * norm
```

a. min为1

Validation Accuracy 0.7233

Valid AUC: 0.8968054921631274

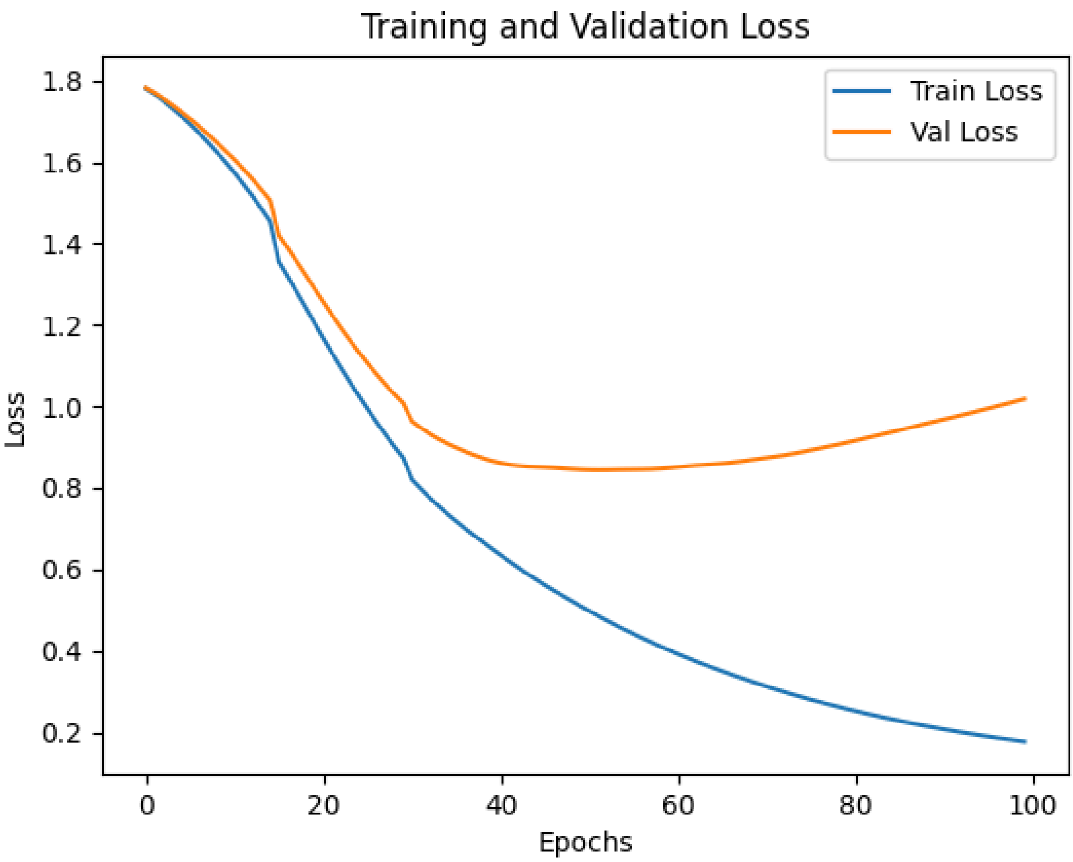


可以看出过拟合得到较好抑制，但是结果有所下降. 可能是因为min值设置过小，导致归一化过于严格，因此调整min值为2

b. min为2

Validation Accuracy 0.7368

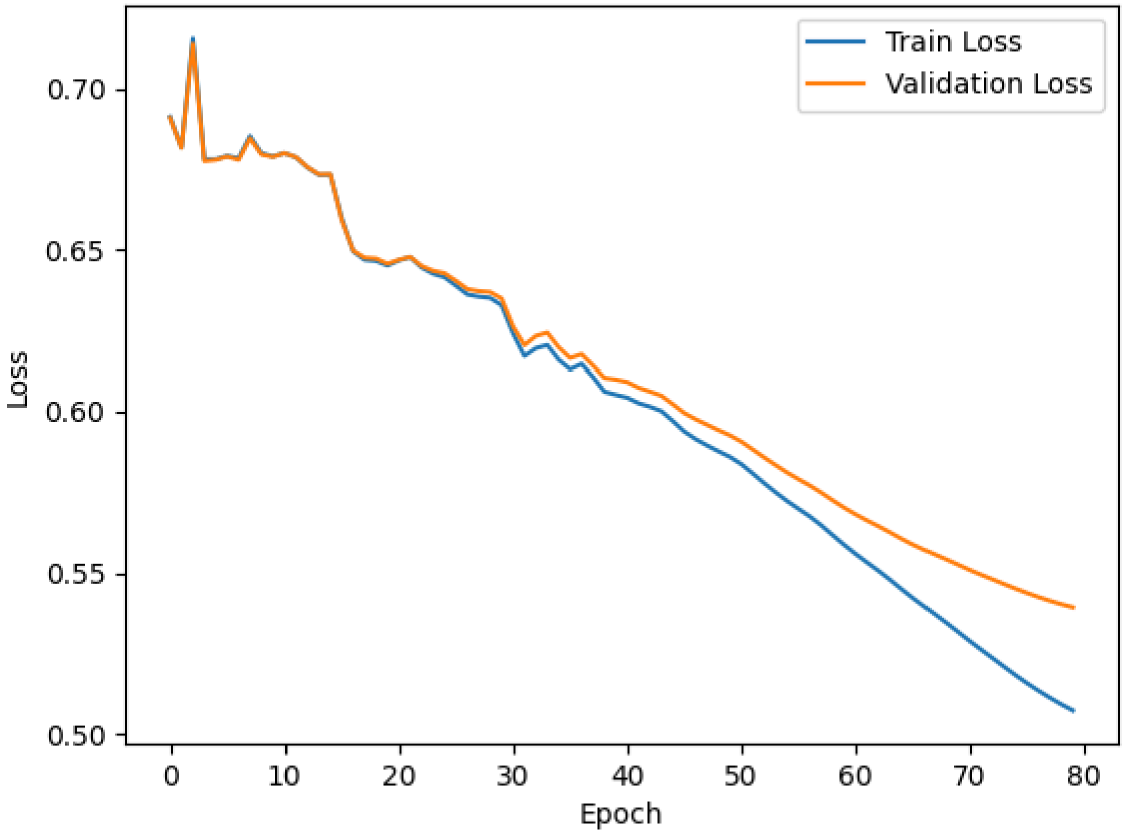
Valid AUC: 0.9063669002122143



可以看出，pairnorm的效果较好，但是效果不如添加自环

> 结论：虽然归一化降低过拟合，提升泛化性，但是也降低了模型的学习能力，进而使得学习率下降

5. 激活函数
   i. relu
      Validation Accuracy 0.7233
      Valid AUC: 0.8968054921631274
   ii. elu
      Validation Accuracy 0.7278
      Valid AUC: 0.9153824798490503
   iii. leaky_relu
      Validation Accuracy 0.7263
      Valid AUC: 0.9104088537132802
   iv. sigmoid
      Validation Accuracy 0.7459
      Valid AUC: 0.9002216251570433



综合来看，虽然总体效果相近，sigmoid函数的效果最好，而且过拟合现象很小

测试结果

citeseer:

节点分类

Test Accuracy 0.7733

链路预测

Test AUC: 0.8545496680328062

cora

节点分类

Test Accuracy 0.8619

链路预测

Test AUC: 0.860964488668269