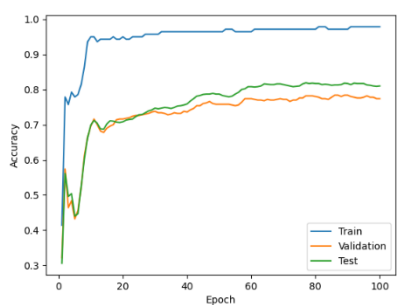
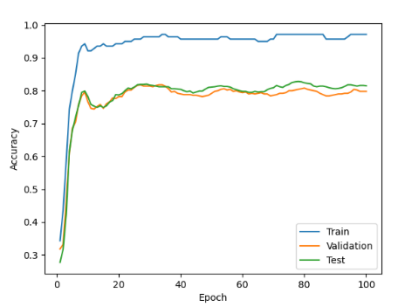
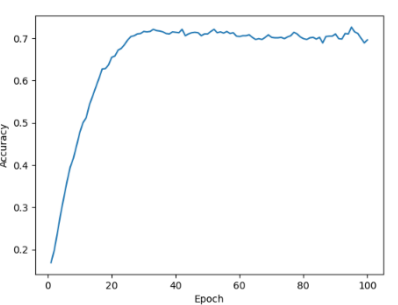


Part One: Graph Neural Networks (GNN) and Node2Vec

1. Task A: GCN, GAT and Node2Vec

GCN	GAT	Node2Vec
		
Train: 0.9786, Val: 0.7740, Test: 0.8100	Train: 0.9714, Val: 0.7980, Test: 0.8150	Acc: 0.7020

结果显示,GCN 和 GAT 在测试集上的准确率都超过了 80%,且两个模型的收敛速度都比较快。Node2Vec 收敛速度相对较慢,模型的准确率在 70%左右。从训练之后的 Node2Vec 模型所生成的 t-SNE 图也表明,该方法可以对 Cora 数据集进行良好的分类。



2. Task B: GIN

$$h^{(k+1)}(v) = \text{MLP}_\Phi \left((1 + \epsilon) \cdot \text{MLP}_f \left(h^{(k)}(v) \right) + \sum_{u \in N(v)} \text{MLP}_f \left(h^{(k)}(u) \right) \right)$$

Different MLPs

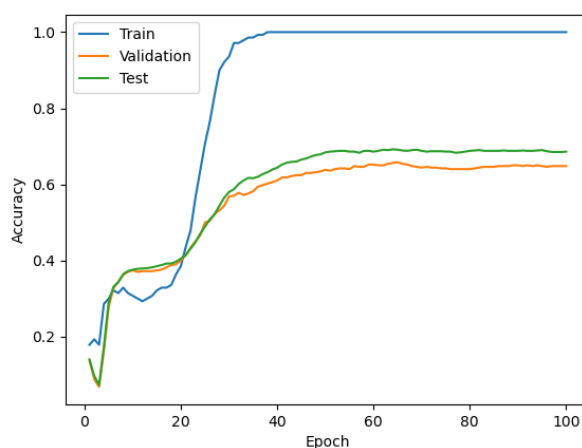
利用 torch_geometric.nn 中的 GINConv 模块,搭建 GIN 模块。

网络参数: GIN 网络的层数(即 MLP_Φ 的层数) num_layers=4;

内层 MLP 网络的层数(即 MLP_f 的层数) num_mlp_layers=2;

MLP 网络中隐藏层的维度 hidden_dim=64。

Performance:

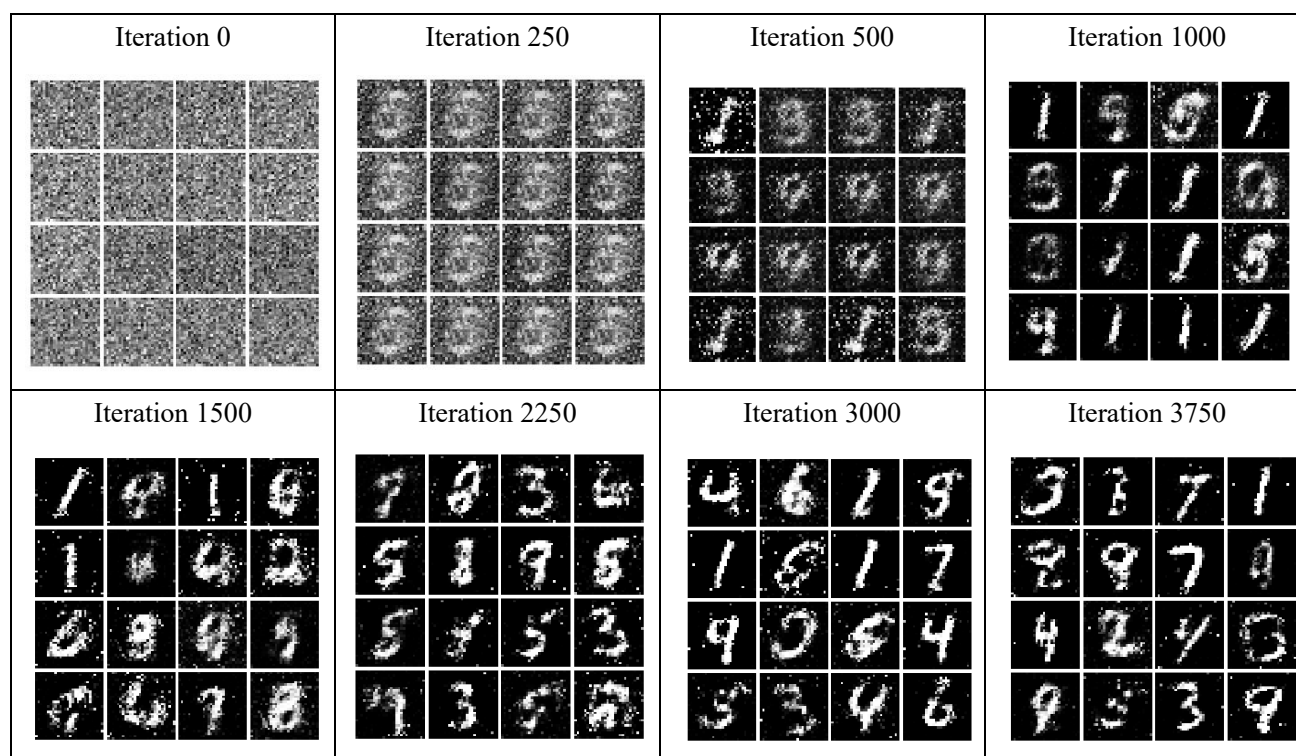


Epoch: 100, Train: 1.0000, Val: 0.6480, Test: 0.6860

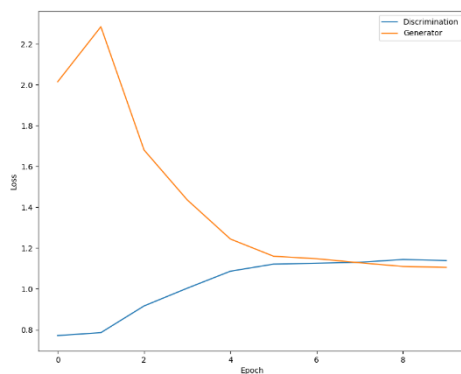
GIN 模型在训练集的准确率非常高，但在测试集上的误差只有 68%，模型存在一定的过拟合现象。

Part Two: Generative Adversarial Networks (GAN)

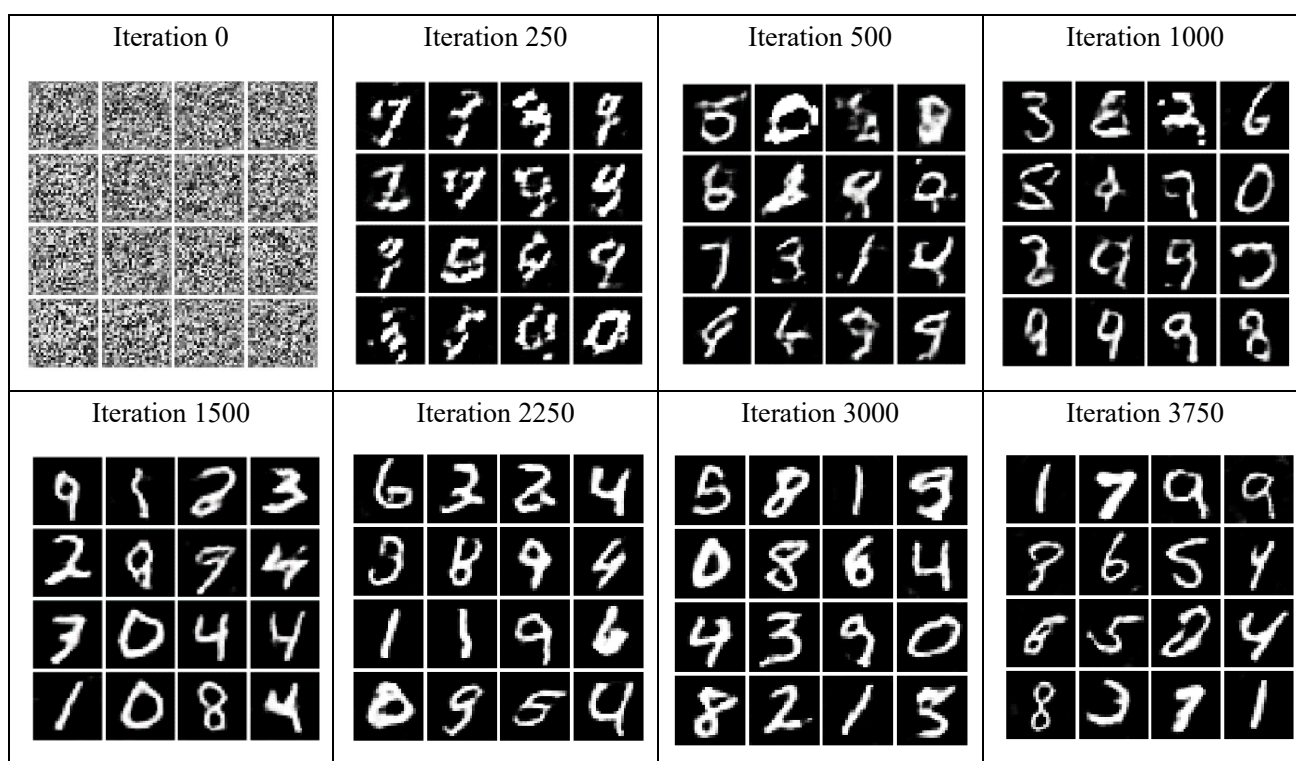
1. GAN Implementation



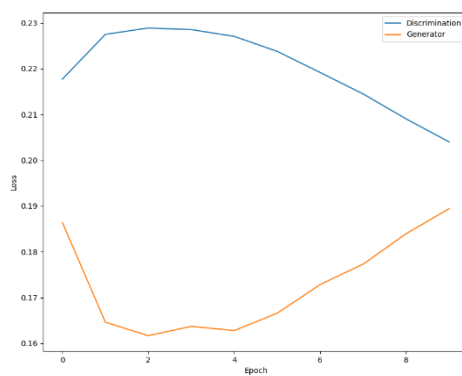
训练过程中，生成器和判别器的 Loss 随 Epochs 的变化如下图所示：



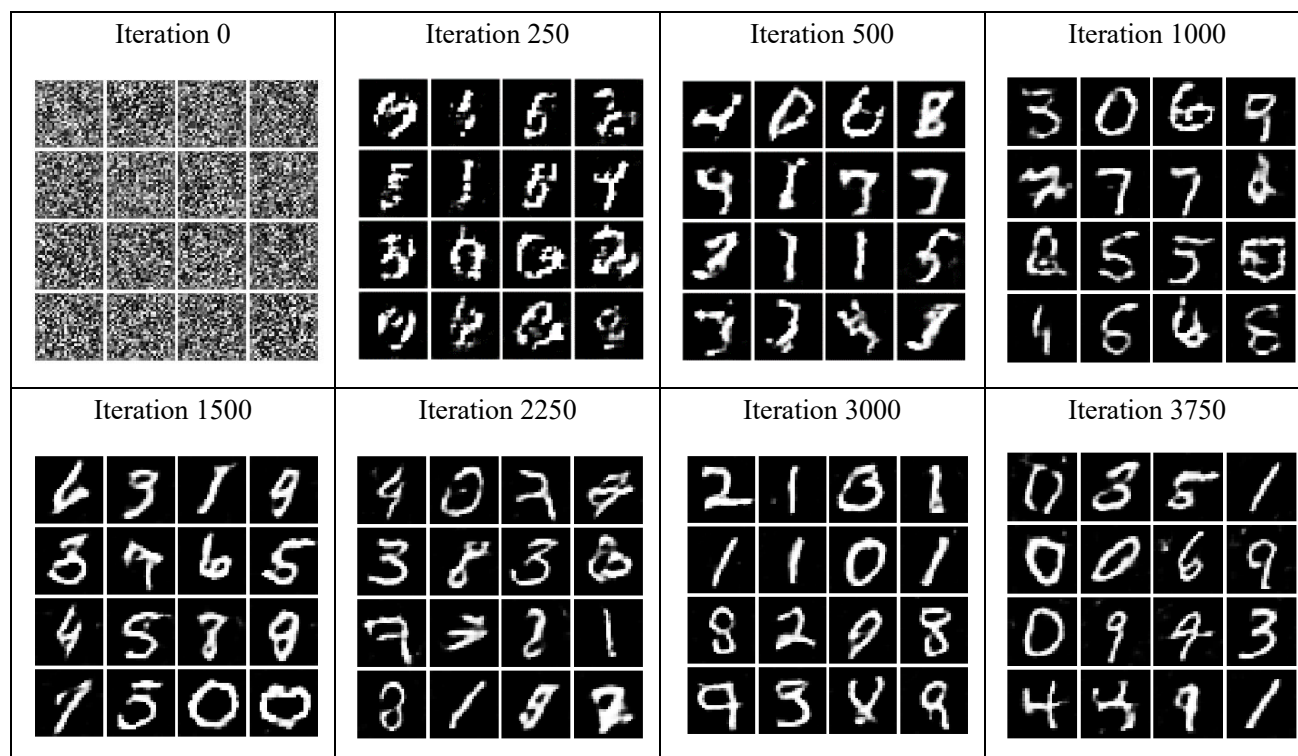
2. Least Squire GAN



训练过程中，生成器和判别器的 Loss 随 Epochs 的变化如下图所示：



3. Deeply Convolutional GAN



训练过程中，生成器和判别器的 Loss 随 Epochs 的变化如下图所示：

