

## 3 GCN 模型（Graph Convolutional Network）

### 3.1 基本原理

#### 3.1.1 背景

神经网络家族里，BP 神经网络是处理表格数据的高手；CNN 在图像处理上独领风骚；RNN 则对序列数据情有独钟。但面对图数据，它们就有点力不从心了。图数据里，节点间的复杂拓扑关系和空间依赖性，传统神经网络难以把握。

在真实世界中，我们会发现很多数据其实是以图的形式存在的，所以处理了图数据就十分重要了。为应对图数据，GNN 横空出世。它能利用图的拓扑结构，通过卷积等操作学习节点低维表示，捕获空间依赖关系。在人脸识别、社交网络分析、电力网络优化等任务上，GCN 比传统神经网络更具优势。

#### 3.1.2 传统神经网络——以 CNN 为例

卷积神经网络（CNN）就像是“图像专家”，它的输入通常是具有欧几里得结构的图片。想象一下，我们有一张照片，CNN 就像是拿着一个放大镜（卷积核或 kernel）在照片上一点一点地移动，通过这个小窗口来仔细观察和提取照片中的特征。因为照片的结构比较规则，所以这个放大镜可以平移着到处看，这就是 CNN 的核心操作。而且，CNN 所处理的照片有一个很特别的地方，那就是它的平移不变性，也就是说，这个放大镜不管移动到照片的哪个角落，看到的局部结构都是一样的。这让 CNN 能够实现参数共享，就像是用同一把尺子去量不同地方的长度，这把尺子（参数）在各个地方都能用，这就是 CNN 的精髓所在。

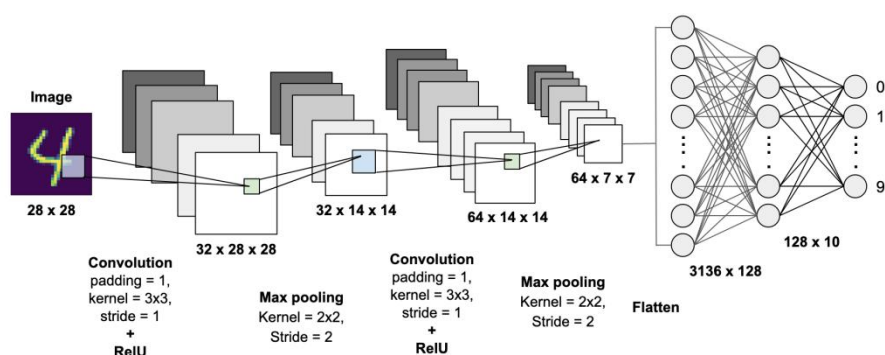
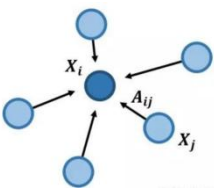


图1 CNN 示意图<sup>1</sup>

<sup>1</sup> 图源互联网

不过，有时候我们会遇到一些结构不那么整齐的网络，这些网络的拓扑结构就像是被风吹乱的蜘蛛网，节点的数量各不相同，每个节点的邻点也各不一样，而且图中的每个节点之间通常都有联系。这让 CNN 有点手足无措，因为它的放大镜不知道该怎么在这样的结构上移动。这时候，图卷积网络（GCN）就出现了，它专门来解决这类问题，让神经网络也能在复杂的网络中大显身手。<sup>[6]</sup>



### 3.1.3 图卷积网络（GCN）

图卷积网络（GCN）是一种专为处理图结构数据而设计的神经网络模型。它与传统的卷积神经网络（CNN）存在显著差异。CNN 适用于图像等具有欧几里得结构的数据，通过在规则的网格上滑动卷积核来提取特征，得益于图像的平移不变性，能够实现参数共享，高效地处理数据。

然而，现实世界中的许多数据并不具备这种规则的欧几里得结构，而是呈现出复杂的图结构。这些图数据由节点和边组成，节点数量不一，每个节点的邻居也各不相同，节点之间的联系错综复杂。在这种情况下，传统的 CNN 难以直接应用，因为它的卷积操作无法在不规则的图结构上有效进行。

图卷积网络（GCN）应运而生，它能够直接在图上进行卷积操作，适用于非欧几里得数据结构。GCN 通过特定的机制，如聚合邻居节点的信息等，来学习节点的表示，从而捕获图中的拓扑结构和节点特征。这使得 GCN 在处理图结构数据时，能够充分发挥其优势，为解决图相关的复杂问题提供了有力的工具。

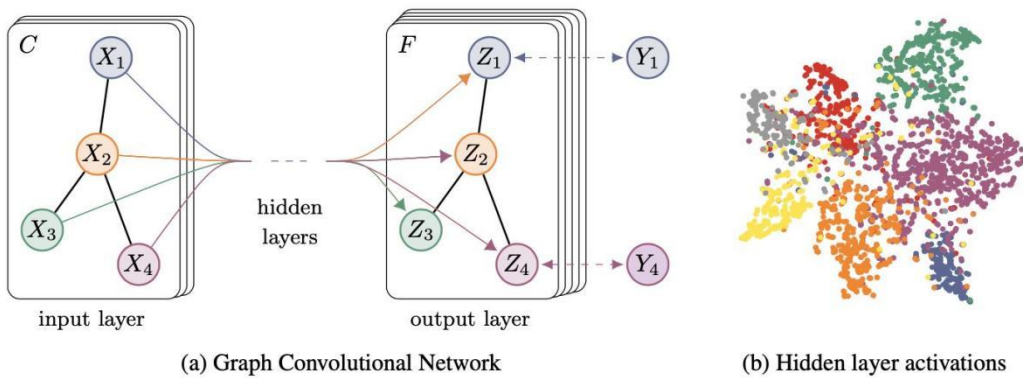


图 2 GCN 结构图<sup>2</sup>

### 3.1.4 图卷积网络（GCN）的类型

图卷积网络（GCN）分为两大类：基于谱的 GCN 和基于空间的 GCN。<sup>[7]</sup>

<sup>2</sup> 图源互联网

1. **基于谱的 GCN:** 这类方法在谱域中定义，主要使用图拉普拉斯矩阵和傅里叶变换。卷积操作通过在谱域中乘以滤波器来实现，利用图拉普拉斯矩阵的特征值和特征向量来执行卷积。这种方法从图信号处理的角度引入滤波器来定义图卷积，将卷积运算解释为从图信号中去除噪声。
2. **基于空间的 GCN:** 这类方法直接在每个节点的连接关系上定义卷积操作，与传统卷积神经网络中的卷积更为相似。代表性方法包括 Message Passing Neural Networks (MPNN)、GraphSAGE 和 Diffusion Convolution Neural Networks (DCNN)等。这些方法通过聚合邻居节点的信息来更新节点的特征表示，从而捕获图中的局部结构信息。

### 3.1.5 图卷积网络 (GCN) 的架构

GCN 通常由多个层组成，每一层负责通过聚合邻居节点的信息来细化节点的嵌入表示。具体架构如下：

- 输入层：初始化节点特征，通常来自原始数据或预训练的嵌入。
- 隐藏层：执行图卷积操作，逐步聚合和转换节点特征。
  - 图卷积层：这些层在图上执行卷积操作，通过聚合邻居节点的特征来更新每个节点的特征表示。
  - 激活函数：如 ReLU 等非线性函数，用于引入模型的非线性。
  - 池化层：通过合并节点来减少图的维度，帮助捕获层次结构。
- 输出层：生成最终的节点嵌入或预测结果，具体取决于任务（例如节点分类、链接预测）。
- 全连接层：在网络的末尾用于执行分类或回归等任务。

### 3.1.6 图卷积网络 (GCN) 的工作原理

GCN 的思想是对于每个节点，考虑其所有邻居以及自身所包含的特征信息。

假设我们手头有一批图数据，其中有 $N$ 个节点 (node), 每个节点都有自己的特征，我们设这些节点的特征组成一个 $N \times D$ 维的矩阵 $X$ , 然后各个节点之间的关系也会形成一个 $N \times N$ 维的矩阵 $A$ , 也称为 邻接矩阵(adjacency matrix)。 $X$ 和 $A$ 便是我们模型的输入。 GCN 也是一个神经网络层，它的层与层之间的传播方式是：

$$H^{(l+1)} = \sigma \left( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

其中：

- $\tilde{A} = A + I$ ,  $I$  是单位矩阵；
- $\tilde{D}$  是  $\tilde{A}$  的度矩阵 (degree matrix), 公式为  $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$  ；
- $H$  是每一层的特征，对于输入层的话， $H$  就是  $X$
- $\sigma$  是非线性激活函数

### 3.1.6 图卷积网络（GCN）的优势和局限性

#### 一、优势：

1. 特征提取：能够有效提取图结构数据的特征，适用于节点分类、图分类和链接预测等任务。
2. 端对端训练：不需要手动定义规则，模型可以自动学习特征和结构信息。
3. 层级结构：通过多层卷积操作，可以捕获不同层次的特征表示。

#### 二、局限性：

1. 层数限制：随着网络层数的增加，节点的嵌入表示可能会变得相似，导致过平滑问题。
2. 可扩展性：大多数图神经网络在处理大型图时效率较低。

## 3.2 算法流程

### 1. 数据预处理：

将原始二进制数据转换为 JPG 图像，再提取人脸关键点并保存为.npy 文件，将标签添加文件中并保存为新的.npy 文件，最后将.npy 文件转化为.pkl 文件。

### 2. 模型构建：

定义 GCN 模型，包含三个图卷积层和一个全连接层。图卷积层用于提取节点特征，全连接层用于输出图的分类结果。

### 3. 训练过程：

- (1) 加载图数据并创建数据加载器。
- (2) 实例化模型、损失函数、优化器和学习率调度器。

- (3) 进行多轮训练，在每轮中遍历数据加载器中的每个批次，执行前向传播、计算损失、反向传播和参数更新。
  - (4) 每隔一定轮数保存模型，并根据早停法判断是否提前结束训练。
4. 测试过程：在测试数据上评估模型性能，计算准确率。

### 3.3 GCN 模型的实现<sup>[8]</sup>

#### 3.3.1 傅里叶变换

傅里叶变换是一种将信号从时域（或空域）转换到频域的数学工具。在图神经网络中，它用于将图信号（节点特征）从图域转换到频域。对于图信号  $x$ ，其傅里叶变换定义为：

$$\hat{x} = U^T x$$

其中， $U$  是图的拉普拉斯矩阵的特征矩阵， $\hat{x}$  是  $x$  在频域上的表示。

#### 3.3.2 卷积

在频域中，卷积操作可以通过点乘实现。对于图信号  $x$  和滤波器  $g$ ，其卷积定义为：

$$g * x = U g(\Lambda) U^T x$$

其中， $g(\Lambda)$  是滤波器在频域上的表示， $\Lambda$  是拉普拉斯矩阵的特征值对角矩阵。这种卷积操作是基于图的频域特性进行的，能够捕捉图中的空间依赖关系。

#### 3.3.3 特征矩阵 $U$

特征矩阵  $U$  是拉普拉斯矩阵  $L$  的特征向量矩阵。拉普拉斯矩阵  $L$  定义为  $L = D - A$ ，其中  $D$  是度矩阵， $A$  是邻接矩阵。特征矩阵  $U$  的每一列对应于拉普拉斯矩阵的一个特征向量，这些特征向量构成了图的频域基。

#### 3.3.4 拉普拉斯算子与拉普拉斯矩阵

拉普拉斯算子在图中用于衡量节点与其邻居之间的差异。拉普拉斯矩阵  $L$  是拉普拉斯算子在图上的离散形式，它是一个对称半正定矩阵。拉普拉斯矩阵的特征值和特征向量具有重要的物理意义，特征值反映了图的频率特性，特征向量则构成了图的频域空间。

#### 3.3.5 Spectral CNN

Spectral CNN 是基于图信号的频域表示进行卷积操作的神经网络模型。它首先将图信号通过傅里叶变换转换到频域，然后在频域上应用滤波器进行卷积操作，最后通过逆傅里叶变换将结果转换回图域。Spectral CNN 的关键在于设计合适的滤波器  $g(\Lambda)$  来捕捉图中的特征。

### 3.3.6 基于多项式的 GCN

为了简化 Spectral CNN 中的滤波器设计，基于多项式的 GCN 提出了一种多项式滤波器。这种滤波器可以表示为拉普拉斯矩阵的多项式：

$$g(\Lambda) = \theta_0 I + \theta_1 \Lambda + \theta_2 \Lambda^2 + \dots + \theta_K \Lambda^K$$

其中， $\theta_K$  是多项式的系数， $I$  是单位矩阵。通过这种方式，滤波器可以更灵活地捕捉图中的不同频率成分。

### 3.3.7 基于切比雪夫多项式的 GCN

切比雪夫多项式是一种特殊的多项式，具有良好的数值稳定性和计算效率。基于切比雪夫多项式的 GCN 利用切比雪夫多项式来近似滤波器，其形式为：

$$g(\Lambda) = \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda})$$

其中， $T_k$  是切比雪夫多项式， $\Lambda$  是归一化的拉普拉斯矩阵的特征值。这种近似方法可以大大减少计算量，同时保持较好的滤波效果。

### 3.3.8 一阶切比雪夫逼近

一阶切比雪夫逼近是一种简化版的基于切比雪夫多项式的 GCN。它只使用一阶切比雪夫多项式来近似滤波器，即：

$$g(\Lambda) = \theta_0 I + \theta_1 \tilde{L}$$

其中， $\tilde{L}$  是归一化的拉普拉斯矩阵。这种逼近方法进一步简化了计算，同时仍然能够捕捉图中的主要特征。一阶切比雪夫逼近是 GCN 中常用的一种形式，因为它在计算效率和模型性能之间取得了较好的平衡。

在本次实验中，使用的是基于一阶切比雪夫逼近的 GCN 方法。具体体现在模型定义部分，即 `GraphConvolutionalNetwork` 类中使用的 `GCNConv` 层。

`GCNConv` 层是 PyTorch Geometric 库中实现的图卷积层，其背后采用的正是类似于一阶切比雪夫逼近的机制来近似频域上的卷积操作。

在一阶切比雪夫逼近中，滤波器  $g(\Lambda)$  表示为

$$g(\Lambda) = \theta_0 I + \theta_1 \tilde{L}$$

其中， $\tilde{L}$  是归一化的拉普拉斯矩阵。

而在 `GCNConv` 层中，其卷积操作可以表示为：

$$\tilde{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)})$$

这里， $\tilde{A}$ 是添加自环后的邻接矩阵， $\tilde{D}$  是对应的度矩阵， $\sigma$  是非线性激活函数， $W^{(l)}$  是第 1 层的可训练权重矩阵。这种形式与一阶切比雪夫逼近的思想相契合，都是通过简化频域上的操作来高效地实现图卷积，从而在计算效率和模型性能之间取得平衡。因此，可以认为程序中使用的是基于一阶切比雪夫逼近的 GCN 方法。

### 3.4 优化算法

- 优化器：选用 Adam 优化器，其初始学习率设定为 0.002。Adam 优化器结合了 RMSprop 和 Momentum 两种优化算法的优点，能够根据参数梯度的大小自动调整学习率，对于不同参数采用不同的学习率更新策略，从而在训练过程中具有较快的收敛速度和较好的稳定性，适用于本实验中 GCN 模型的参数优化。
- 学习率调度：自定义学习率调度函数，前 400 轮训练中学习率保持恒定不变，为模型初期的快速收敛提供支持；随后 200 轮训练里，学习率会逐步递减，使得模型在接近最优解时能够更加精细地调整参数，避免因学习率过大导致的震荡；当训练轮数超过 600 轮后，学习率固定在 0.0001，以维持模型参数的稳定性。通过 LambdaLR 学习率调度器实现该调度策略，根据当前轮数动态调整优化器的学习率。一开始设置学习率可以降到 0，但发现这会导致在结束前衰减到 0，导致后半段训练毫无成效。因此设置了最低学习率和早停。
- 损失函数：采用 CrossEntropyLoss 作为损失函数。由于本实验的输出是两个类别的 logits 值（即性别分类的两个类别对应的得分），CrossEntropyLoss 能够自动对输出进行 softmax 归一化处理，将 logits 值转换为概率分布，并计算模型预测概率分布与真实标签概率分布之间的交叉熵损失。该损失函数广泛应用于多分类问题，在本实验中可有效衡量模型预测结果与真实标签之间的差异，为模型训练提供明确的优化方向。
- 早停法：为了防止模型在训练过程中出现过拟合现象，即模型在训练集上的性能不断提升，但在测试集上的性能却开始下降，本实验引入了早停法（Early Stopping）。具体实现方式为：从第 200 轮训练开始启用早停法，设置耐心

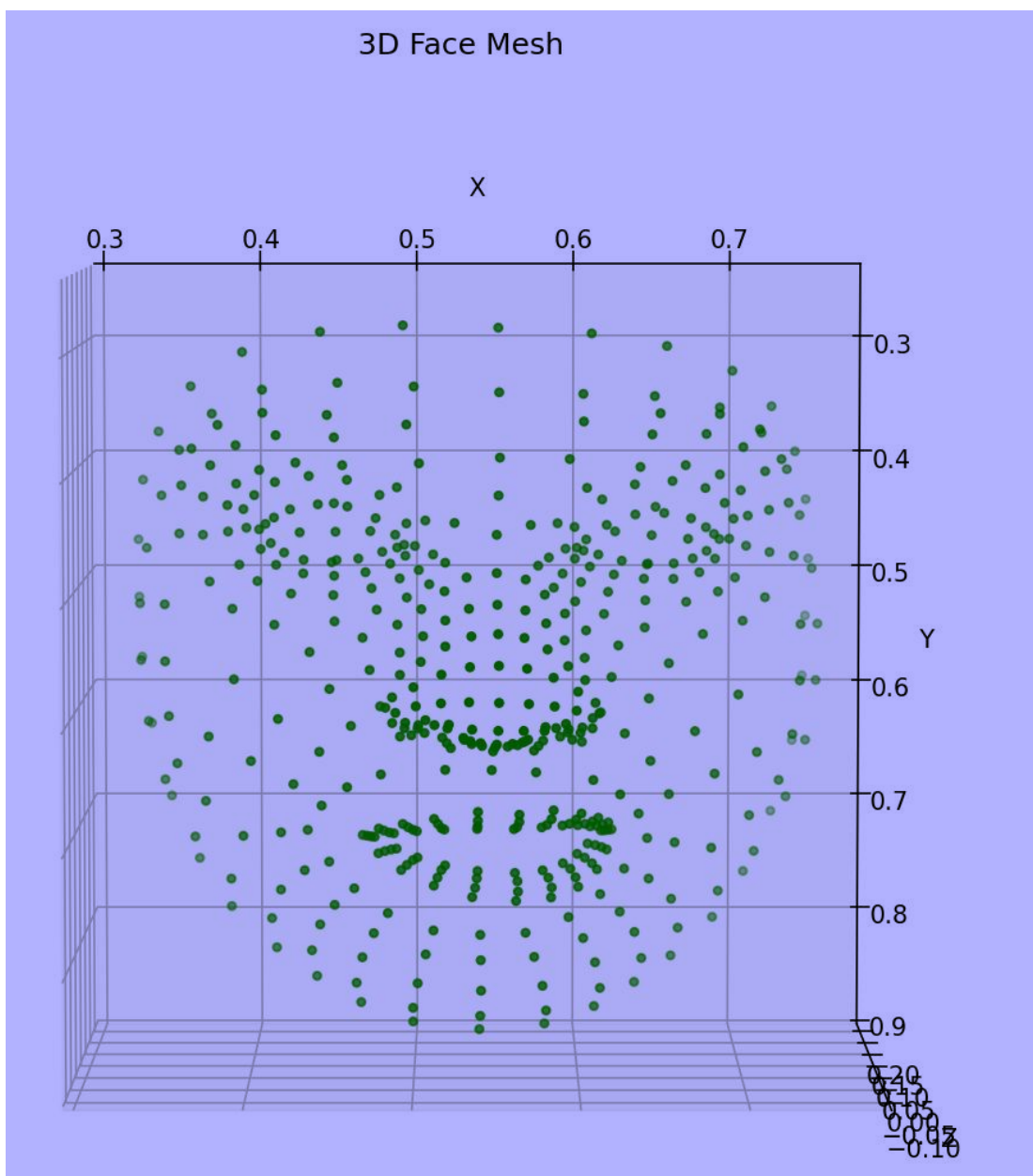
值（`early_stopping_patience`）为 80。在训练过程中，每当完成一轮训练后，都会在测试集上评估模型的准确率。如果当前轮次的测试准确率高于之前记录的最佳准确率（`best_accuracy`），则更新最佳准确率并将耐心值计数器（`patience_counter`）重置为 0，同时保存当前模型为最佳模型；反之，若测试准确率未超过最佳准确率，则耐心值计数器加 1。一旦耐心值计数器达到 80，即在连续 80 轮训练中模型测试准确率均未得到改善，便提前终止训练。由于模型数据过大，在初期因为数值设计不合理，很多次模型未能达到理想情况就因为早停法而停下导致浪费了运算资源。因此，合理设置早停法，通过在合适的时机停止训练，使得模型能够在保持较好泛化能力的前提下，避免因过度训练而导致的性能下降，从而提高模型在实际应用中的有效性和可靠性。

### 3.5 数据处理

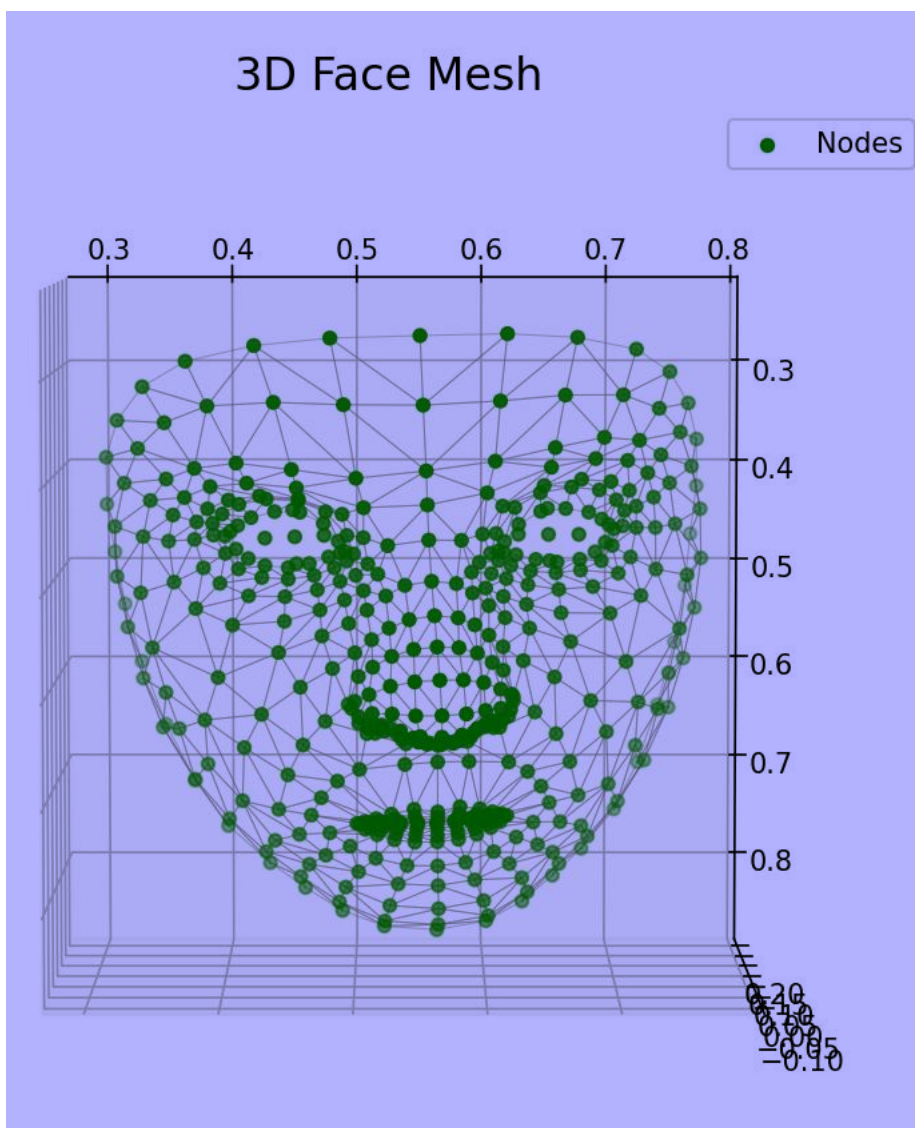
#### 1. 使用 `output_jpg.py` 程序将原始数据生成.jpg 格式；

将存储为原始二进制格式的人脸图像数据转换为.jpg 格式的图像文件。这些原始数据文件位于指定的目录中，每个文件包含 128x128 像素的灰度图像数据。通过读取二进制数据，将其重塑为 128x128 的数组，并使用 PIL 库将其转换为.jpg 格式，保存到输出目录中。





2. 使用 `jpg_3np.py` 程序将生成的.jpg 格式数据转化成  $3 \times 478$  的.npy 文件；  
使用 `Mediapipe Face Mesh` 模块处理.jpg 图像，提取人脸关键点。Face Mesh 能够检测到人脸上的 478 个关键点的 3D 坐标。将这些关键点坐标保存为.npy 文件，每个.npy 文件包含一个形状为  $(3, 478)$  的数组，其中每一行分别代表 x、y、z 坐标。



3. 使用 `csvbuild.py` 程序提取原始数据中各个人像的性别数据生成.csv 文件;
4. 使用 `3npv_4npv.py` 程序将  $3 \times 478$  的.npy 文件转化为  $4 \times 478$  的.npy 文件;  
从 CSV 文件中读取标签信息,并将其添加到对应的.npy 文件中。CSV 文件中每一行包含一个索引和一个标签(0 或 1,表示性别)。将标签作为新的行添加到.npy 文件,形成一个形状为  $(4, 478)$  的数组,其中第四行是标签信息。
5. 使用 `makepkl.py` 程序将所有的  $4 \times 478$  的.npy 文件随机抽取 1600 份转化为.pkl 文件;

将包含人脸关键点和标签的.npy 文件转换为图数据,并保存为.pkl 文件。使用 PyTorch Geometric 库构建图数据,其中节点特征是人脸关键点 3D 坐标,边是根据 Face Mesh 的拓扑结构定义的。标签信息作为图的标签。这些.pkl 文件将用于后续的图卷积网络训练。

6. 使用 `test_train.py` 程序将 1600 份 `.pkl` 文件随机抽取 150 份和 1200 份分别存储到 `test` 文件夹和 `train` 文件夹；

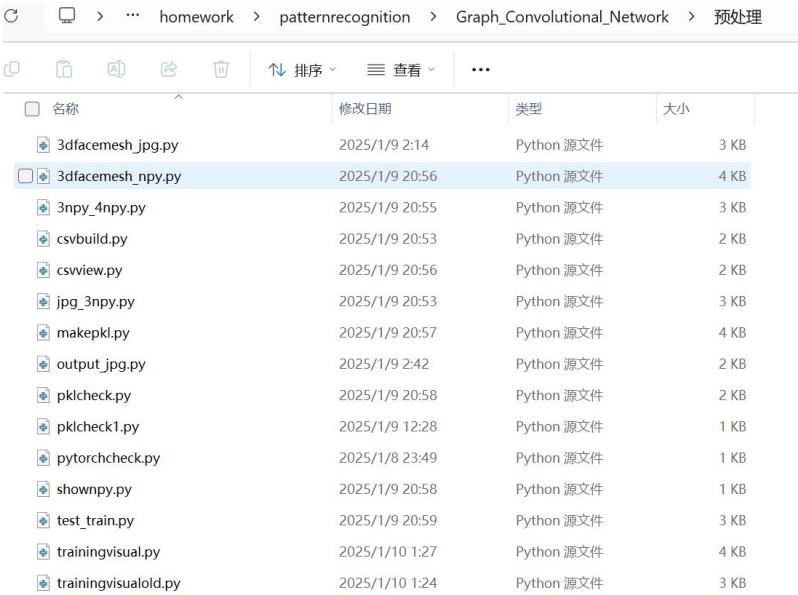
至此，数据预处理已经完成。

其他程序功能

1. 可以使用 `csvview.py`、`pklcheck.py`、`pklchek1.py`、`shownpy.py` 等程序查看生成的各类预处理文件。

2. 其中，`trainingvisual.py`、`trainingvisualold.py` 为模型准确率等数据和轮次关系的可视化，`3dfacemesh_jpg.py`、`3dfacemesh_npy.py` 是面部 3d 模型的可视化。

3. `pytorchcheck.py` 可以用来检测系统 GPU、`pytorch` 等必备库的版本。



名称	修改日期	类型	大小
3dfacemesh_jpg.py	2025/1/9 2:14	Python 源文件	3 KB
3dfacemesh_npy.py	2025/1/9 20:56	Python 源文件	4 KB
3npv_4npv.py	2025/1/9 20:55	Python 源文件	3 KB
csvbuild.py	2025/1/9 20:53	Python 源文件	2 KB
csvview.py	2025/1/9 20:56	Python 源文件	2 KB
jpg_3npv.py	2025/1/9 20:53	Python 源文件	3 KB
makepkl.py	2025/1/9 20:57	Python 源文件	4 KB
output_jpg.py	2025/1/9 2:42	Python 源文件	2 KB
pklcheck.py	2025/1/9 20:58	Python 源文件	2 KB
pklchek1.py	2025/1/9 12:28	Python 源文件	1 KB
pytorchcheck.py	2025/1/8 23:49	Python 源文件	1 KB
shownpy.py	2025/1/9 20:58	Python 源文件	1 KB
test_train.py	2025/1/9 20:59	Python 源文件	3 KB
trainingvisual.py	2025/1/10 1:27	Python 源文件	4 KB
trainingvisualold.py	2025/1/10 1:24	Python 源文件	3 KB

## 3.6 参数设置

### 3.6.1 参数一

如下图所示：

```

from torch.nn import Linear, Conv2d, BatchNorm1d, BatchNorm2d, MaxPool2d
from torch.optim.lr_scheduler import LambdaLR

# 超参数配置
config = {
    'input_dim': 3, # 输入特征维度 (3D 坐标)
    'hidden_dim': 256, # 隐藏层的维度
    'output_dim': 2, # 输出维度 (2: 性别男女)
    'learning_rate': 0.002, # 初始学习率
    'batch_size': 32, # 每个批次的数据量
    'epochs': 600, # 训练的轮数
    'save_interval': 5, # 每几个epoch保存一次模型
    'print_every_sample': 5, # 每几个样本汇报一次效果
    'save_dir': 'D:/桌面/homework/patternrecognition/Graph_Convolutional_Network/saved_models', # 模型保存路径
    'log_file': 'D:/桌面/homework/patternrecognition/Graph_Convolutional_Network/training.txt', # 训练日志文件路径
    'threshold': 0.5, # 判断为正样本的置信度
    'lr_stable_epochs': 400, # 固定学习率的轮数
    'lr_decay_epochs': 200, # 学习率衰减的轮数
    'early_stopping_patience': 80, # 早停法的耐心值
    'early_stopping_start_epoch': 200, # 从第200轮开始使用早停法
}

# 确保保存目录存在
os.makedirs(config['save_dir'], exist_ok=True)

```

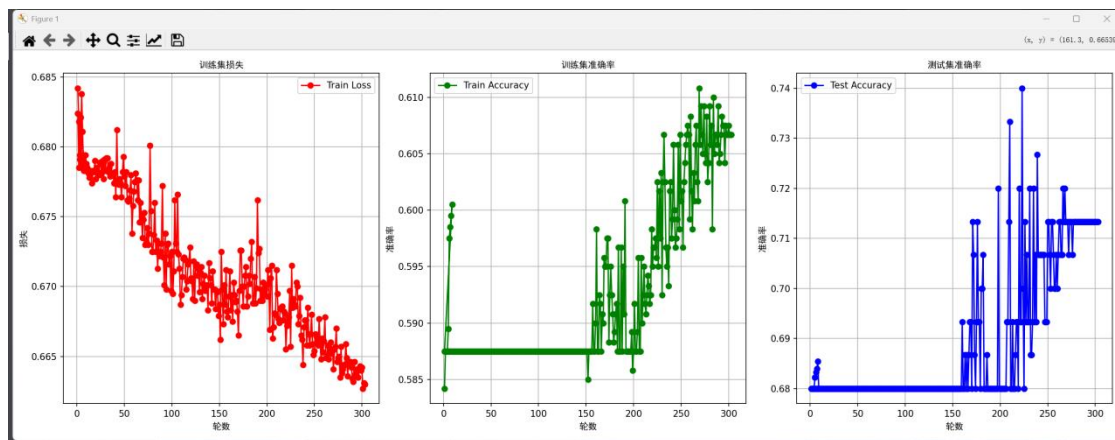
结果如下图：

```

Epoch 298, Loss: 0.6643, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 5.0000000000000004e-05
Epoch 299, Loss: 0.6640, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 3.00000000000000028e-05
Epoch 300, Loss: 0.6642, Train Accuracy: 0.6075, Test Accuracy: 0.7133, lr: 1.0000000000000001e-05
Epoch 301, Loss: 0.6627, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 2.0000000000000002e-07
Epoch 302, Loss: 0.6631, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 2.0000000000000002e-07
Epoch 303, Loss: 0.6630, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 2.0000000000000002e-07

```

应触发早停法结束，最终模型准确率为：0.7133



### 3.6.2 参数二

如下图所示：



```

# --- 超参数配置 ---
config = {
    'input_dim': 3, # 输入特征维度 (3D 坐标)
    'hidden_dim': 512, # 隐藏层的维度
    'output_dim': 2, # 输出维度 (2: 性别男女)
    'learning_rate': 0.002, # 初始学习率
    'batch_size': 32, # 每个批次的的数据量
    'epochs': 600, # 训练的轮数
    'save_interval': 5, # 每几个epoch保存一次模型
    'print_every_sample': 5, # 每几个样本汇报一次效果
    'save_dir': 'D:/桌面/homework/patternrecognition/Graph_Convolutional_Network/saved_models', # 模型保存路径
    'log_file': 'D:/桌面/homework/patternrecognition/Graph_Convolutional_Network/training.txt', # 训练日志文件路径
    'threshold': 0.5, # 判断为正样本的置信度
    'lr_stable_epochs': 340, # 固定学习率的轮数
    'lr_decay_epochs': 260, # 学习率衰减的轮数
    'early_stopping_patience': 80, # 早停法的耐心值
    'early_stopping_start_epoch': 600, # 从第200轮开始使用早停法
}

```

调整了早停法，使模型可以完整训练 600 轮

结果如下图：

```

Epoch 595, Loss: 0.6669, Accuracy: 0.6075, lr: 0.0
Epoch 596, Loss: 0.6658, Accuracy: 0.6075, lr: 0.0
Epoch 597, Loss: 0.6642, Accuracy: 0.6075, lr: 0.0
Epoch 598, Loss: 0.6650, Accuracy: 0.6075, lr: 0.0
Epoch 599, Loss: 0.6664, Accuracy: 0.6075, lr: 0.0

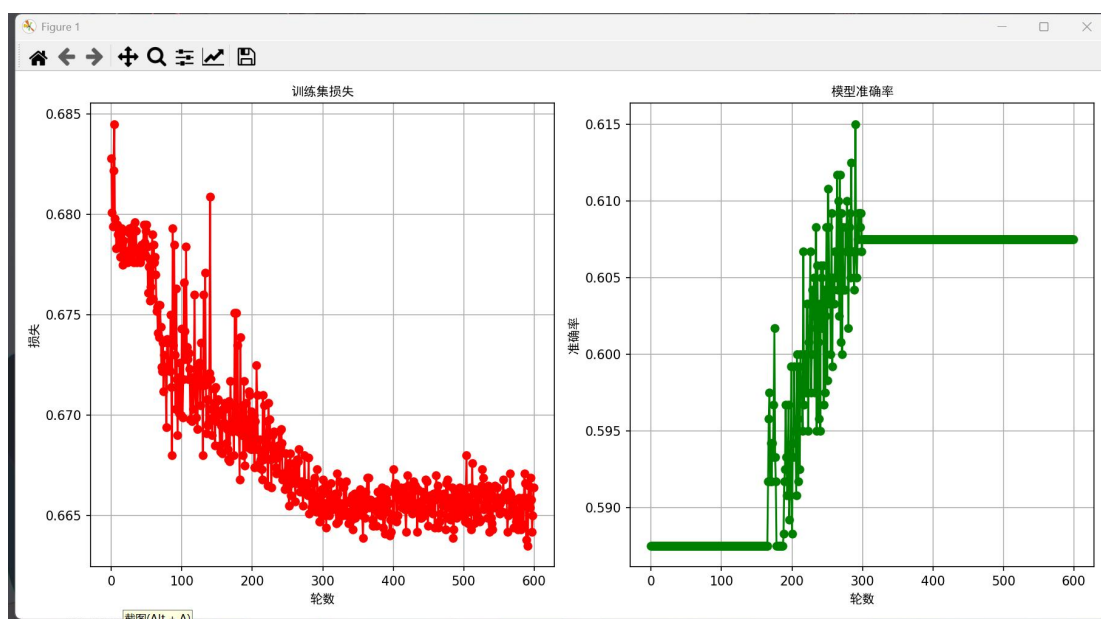
```

学习率衰减至 0，最终模型准确率为：0.7133

```

size mismatch for fc.weight: copying a param with shape torch.Size(
PS D:\桌面\homework\patternrecognition\Graph_Convolutional_Network> & d:/桌
h_Convolutional_Network/test.py
D:\桌面\homework\patternrecognition\Graph_Convolutional_Network\conda\lib\
' instead
warnings.warn(out)
测试准确率为: 0.7133
PS D:\桌面\homework\patternrecognition\Graph_Convolutional_Network> 

```





CrossEntropyLoss 损失函数，并通过自定义的学习率调度策略调整学习率。通过早停法避免了过拟合，提高了模型的泛化能力。最终，模型在测试数据上取得了较高的准确率，验证了 GCN 模型在处理图结构数据方面的有效性。

在与其他算法的对比中，GCN 在处理图数据时表现出色。例如，与传统的图算法（如 PageRank、K-means 等）相比，GCN 具有更强的表达能力和泛化性，因此在本次人脸识别取得了显著成果。然而，GCN 也存在一些挑战，如过拟合问题和计算效率等。

### 3.7.2 需要改进的方面

尽管 GCN 在图数据处理中表现出色，但仍有一些需要改进的地方：

1. 层数限制：随着网络层数的增加，节点的嵌入表示可能会变得相似，导致过平滑问题。为缓解这一问题，可以使用残差连接或跳跃连接（如 GraphSAGE）。
2. 可扩展性：大多数图神经网络在处理大型图时效率较低。为提高可扩展性，可以使用采样方法（如 FastGCN 或 GraphSAGE）来降低计算复杂度。
3. 异构图和动态图的支持：目前的采样算法主要基于静态的同构图进行优化，忽略了现实应用中图数据的异构性、动态性、幂律分布等复杂特征。未来的研究可以考虑开发支持异构图、动态图和多模态图的模型。
4. 结合其他模型：可以将 GCN 与 Transformer 架构相结合，利用 Transformer 的全局建模能力提升 GCN 的表现力。此外，还可以将 GCN 与传统机器学习模型（如 SVM、树模型）结合，解决小数据集或弱监督问题。

## 参考文献

1. Thomas N. Kipf and Max Welling, Semi-Supervised Classification with Graph Convolutional Networks, arXiv, 1609.02907. 2017. <https://arxiv.org/abs/1609.02907.pdf>
2. Zhou J, Cui G, Zhang Z, et al. Graph neural networks: A review of methods and applications[J]. arXiv preprint arXiv:1812.08434, 2018.; <https://arxiv.org/pdf/1812.08434.pdf>
3. 徐冰冰,岑科廷,黄俊杰,沈华伟,程学旗.图卷积神经网络综述[J/OL].计算机学报,2019:1-31.
4. Zhang Z, Cui P, Zhu W. Deep learning on graphs: A survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2020.
5. Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020.

6. 无枵,深入理解图卷积神经网络(GCN)原理,CSDN,2020.

<https://blog.csdn.net/fs1341825137/article/details/109783308>

7. 不务正业的土豆,图卷积网络 GCN Graph Convolutional Network (谱域 GCN)的理解和详细推导,CSDN,2019. <https://blog.csdn.net/yyl424525/article/details/100058264>

8. Semeron, 图神经网络之 GCN 原理、示例及代码实现, 知乎,

<https://zhuanlan.zhihu.com/p/633419078>

## 个人感悟:

在本次实验中,我最初计划采用卷积神经网络(CNN)进行训练,但由于组内已有同学选择了该算法,我转而选择了我相对熟悉的图卷积网络(GCN)。作为集成电路专业的学生,我最初接触 GCN 是因为它在 AMD GPU 架构中的应用,而非图卷积网络。但去年,我在参加集创赛并需要开发一个车辆识别系统时,偶然了解到 GCN 模型,并与队友进行了初步尝试,尽管结果不如 YOLO 算法,但那次经历让我对 GCN 产生了浓厚的兴趣。

通过这次实验,我有机会从零开始编写程序,完成 GCN 模型的训练。在这个过程中,我深刻体会到了理论与实践相结合的重要性。我意识到,尽管 GCN 在某些方面与 CNN 有相似之处,但它在处理图结构数据方面具有独特的优势。GCN 能够捕捉节点间的复杂关系,这对于模式识别任务来说是至关重要的。

然而,我也发现 GCN 在人脸识别任务中的应用似乎并不如预期。在实验过程中,我注意到 GCN 在处理矢量图这类具有明确拓扑结构的数据时表现更佳。这让我意识到,选择合适的模型对于特定任务的成功至关重要。尽管如此,这次实验仍然让我对 GCN 有了更深入的理解,并提高了我的编程和问题解决能力。

在实验过程中,我采用了 Adam 优化器和 CrossEntropyLoss 损失函数,并通过自定义的学习率调度策略调整学习率,以提高模型的训练效率。此外,我还引入了早停法来避免过拟合,从而提高了模型的泛化能力。最终,尽管 GCN 在人脸识别任务上的表现不如预期,但我在测试数据上仍然取得了一定的准确率,这验证了 GCN 在处理图结构数据方面的潜力。

通过这次实验,我学到了许多宝贵的经验。我认识到了在模型选择和算法应用上需要更加谨慎,同时也意识到了深入理解模型原理和调整参数的重要性。未



来，我希望能够进一步探索 GCN 在不同领域的应用，并尝试将其与其他模型相结合，以解决更复杂的模式识别问题。此外，我也期待在处理大型图数据时，能够找到提高 GCN 可扩展性的方法，使其在实际应用中发挥更大的作用。