



廣東工業大學

课 程 报 告

课程名称	模式识别
题目名称	基于人脸面部图像的性别识别
专业班级	集成电路设计与集成系统 22（2）
学号姓名	3122009644 杨纯一
	3122009606 陈柏壮
	3122009607 陈进宇
	3222009648 程韵芝
	3222009650 王莺璇
指导教师	邢延

2025 年 01 月 02 日

目录

1 模式识别系统设计	1
2 GCN 模型 (Graph Convolutional Network)	2
2.1 基本原理.....	2
2.2 算法流程.....	5
2.3 GCN 模型的实现 ^[8]	6
2.4 优化算法.....	8
2.5 数据处理.....	9
2.6 参数设置.....	12
2.7 小结.....	15
3 决策树算法	17
3.1 基本原理.....	17
3.2 关键概念.....	17
3.3 算法流程.....	18
3.4 优化算法.....	20
3.5 参数设置.....	21
3.6 小结.....	22
4 k-近邻算法 (K-Nearest Neighbors, KNN)	23
4.1 经典算法的原理.....	23
4.2 参数设定与调整.....	26
4.3 实验结果的分析比较.....	27
4.4 小结.....	28
5 BP 神经网络	29
5.1 简介.....	29
5.2 参数选择.....	31
5.3 程序设计.....	32
5.4 优化方案.....	38
5.5 实验结果.....	40
5.6 小结.....	40
6 支持向量机	41
6.1 经典算法的原理.....	41
6.2 参数设定.....	44
6.3 实验结果的分析比较.....	48
6.4 小结.....	49
7 结论	50
7.1 算法工作与优缺点比较.....	50
7.2 需要改进的方面.....	52
参考文献.....	53
附录.....	54

1 模式识别系统设计

本项目选择人脸识别中的性别识别作为识别、分类目标，采用 Python 语言编程，采用 Pycharm、VScode 等集成开发环境深入探索不同机器学习算法在模式识别任务中的应用效果。在模式识别领域，人脸识别作为一项关键技术，不仅要求准确区分不同个体，还需要对个体特征进行精细分类，如性别识别。为此，我们选取了图卷积网络（GCN）、决策树、K-近邻算法（KNN）、BP 神经网络和支持向量机（SVM）五种具有代表性的算法进行研究。

我们首先对人脸图像数据进行了详细的预处理工作，包括图像裁剪、灰度化、归一化等步骤，以确保数据质量满足算法要求。同时，为了增强模型的泛化能力，我们还采用了数据增强技术，如旋转、缩放、平移等，以增加训练样本的多样性。

在算法实现方面，我们针对每种算法进行了深入的研究和实现。GCN 算法通过构建人脸图像的图结构，利用卷积操作学习节点特征，实现了对性别的有效识别。决策树算法则通过一系列 if-then 规则对人脸特征进行分类，其直观易懂的决策过程为性别识别提供了有力的支持。KNN 算法通过计算待分类样本与训练集中样本的距离，选择最近的 K 个邻居进行投票，从而实现了高精度的性别识别。BP 神经网络通过构建多层前馈神经网络，利用反向传播算法优化网络参数，实现了对人脸性别的识别。SVM 算法结合特征提取技术，通过寻找最优超平面对人脸特征进行分类，同样取得了令人满意的效果。

在每一算法中，我们使用了准确率、精确度、召回率、F1 分数等多个指标对模型进行评估。以评估结果为基本依据将这些算法在图片数据处理方面的工作特点和优缺点进行了比较并作总结。模式识别作为人工智能领域的一个重要分支，其核心在于从大量数据中提取有用信息，发现数据之间的内在联系和规律。在本项目中，我们通过不同算法对人脸图像的特征提取和分类实现了性别识别，这一过程不仅体现了模式识别的基本原理和方法，也为其他模式识别任务提供了有益的参考和借鉴。

总的来说，本项目通过人脸识别-性别识别任务的研究和实践，不仅验证了不同机器学习算法在模式识别任务中的应用效果，还为提高人脸性别识别的准确性和鲁棒性提供了有益的探索和实践。未来，我们将继续深入研究其他先进的机器

学习算法和模式识别技术，延续模式识别学习。

2 GCN 模型（Graph Convolutional Network）

2.1 基本原理

2.1.1 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。

2.1.2 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。

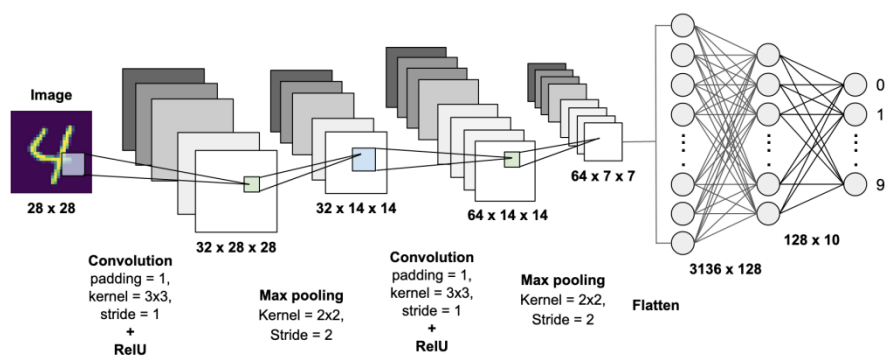
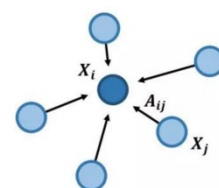


图 1 CNN 示意图¹

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。^[6]



2.1.3 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。

¹ 图源互联网

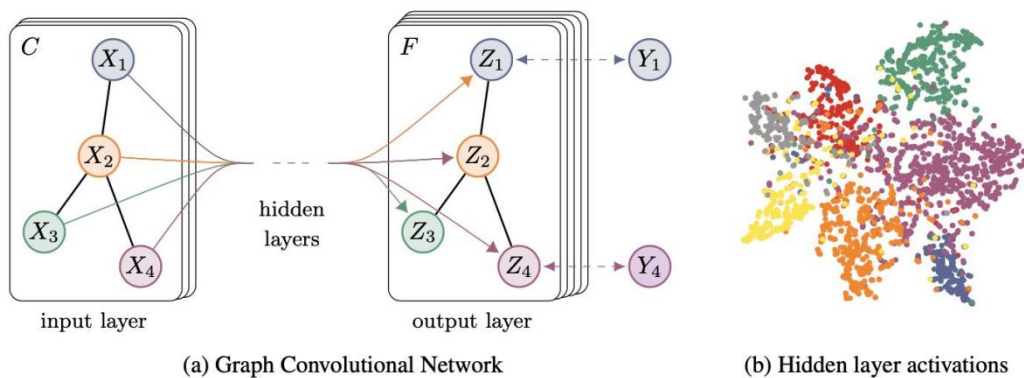


图 2 GCN 结构图²

2.1.4 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。^[7]

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。

2.1.5 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、

² 图源互联网

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。

2.1.6 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
- σ 是非线性激活函数

2.1.6 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。

2.2 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。

2.3 GCN 模型的实现^[8]

2.3.1 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 \mathbf{x} ，其傅里叶变换定义为：

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

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

2.3.2 卷积

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

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

其中， $\mathbf{g}(\Lambda)$ 是滤波器在频域上的表示， Λ 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。

2.3.3 特征矩阵 \mathbf{U}

特征矩阵 \mathbf{U} 是拉普拉斯矩阵 \mathbf{L} 的特征向量矩阵。拉普拉斯矩阵 \mathbf{L} 定义为 $\mathbf{L} = \mathbf{D} - \mathbf{A}$ ，其中 \mathbf{D} 是度矩阵， \mathbf{A} 是邻接矩阵。特征矩阵 \mathbf{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。

2.3.4 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。

2.3.5 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。

2.3.6 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， θ_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。

2.3.7 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， Λ 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。

2.3.8 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， σ 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。

2.4 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。

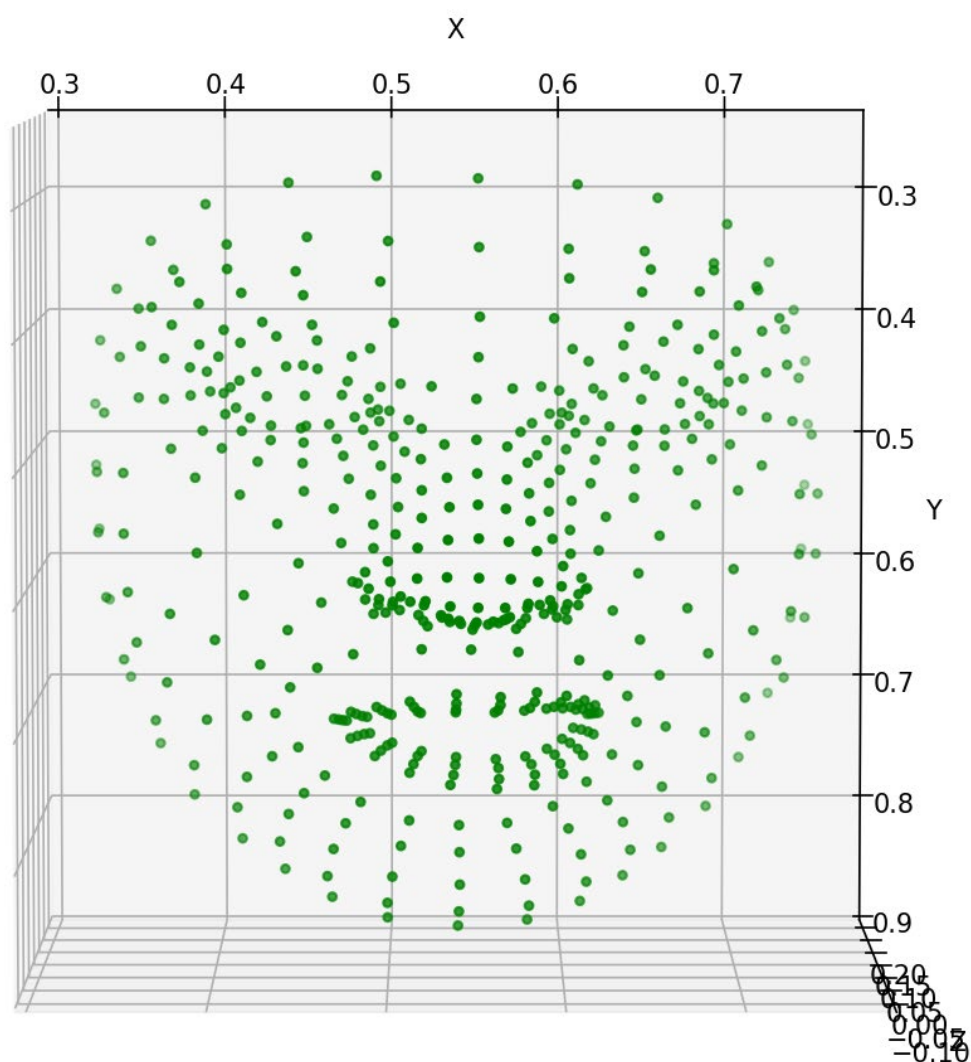
- 损失函数：采用 `CrossEntropyLoss` 作为损失函数。由于本实验的输出是两个类别的 `logits` 值（即性别分类的两个类别对应的得分），`CrossEntropyLoss` 能够自动对输出进行 `softmax` 归一化处理，将 `logits` 值转换为概率分布，并计算模型预测概率分布与真实标签概率分布之间的交叉熵损失。该损失函数广泛应用于多分类问题，在本实验中可有效衡量模型预测结果与真实标签之间的差异，为模型训练提供明确的优化方向。
- 早停法：为了防止模型在训练过程中出现过拟合现象，即模型在训练集上的性能不断提升，但在测试集上的性能却开始下降，本实验引入了早停法（`Early 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。

2.5 数据处理

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

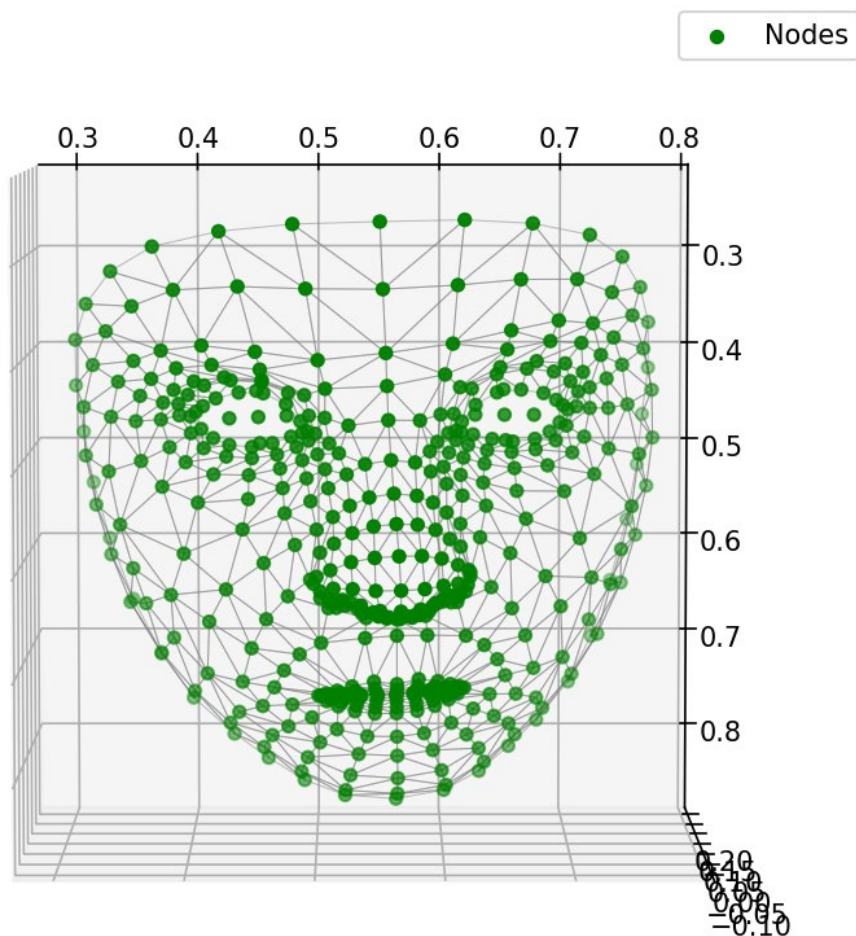
将存储为原始二进制格式的人脸图像数据转换为 `.jpg` 格式的图像文件。这些原始数据文件位于指定的目录中，每个文件包含 128x128 像素的灰度图像数据。通过读取二进制数据，将其重塑为 128x128 的数组，并使用 `PIL` 库将其转换为 `.jpg` 格式，保存到输出目录中。

3D Face Mesh



2. 使用 `jpg_3np.py` 程序将生成的.jpg 格式数据转化成 3×478 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，

3D Face Mesh

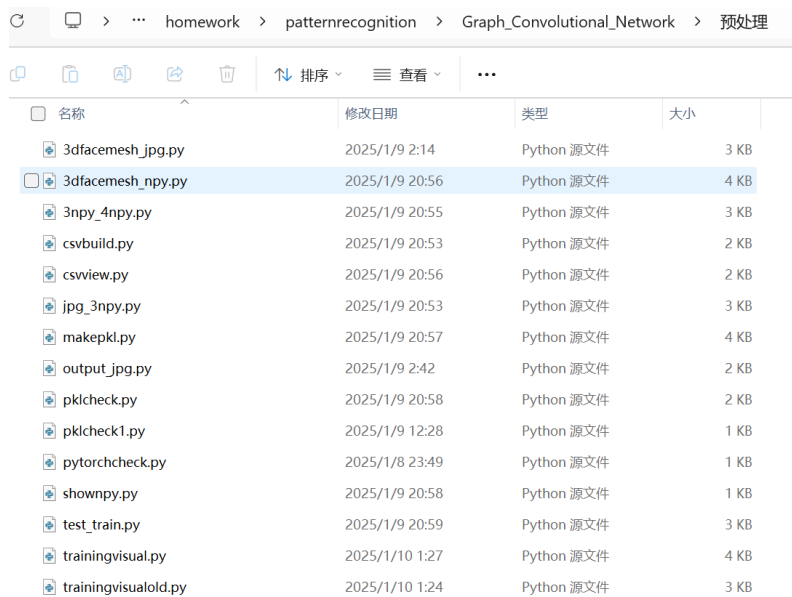


3. 使用 `csvbuild.py` 程序提取原始数据中各个人像的性别数据生成.csv 文件;
4. 使用 `3np4np.py` 程序将 3×478 的.npy 文件转化为 4×478 的.npy 文件;
从 CSV 文件中读取标签信息,并将其添加到对应的.npy 文件中。CSV 文件中每一行包含一个索引和一个标签 (0 或 1, 表示性别)。将标签作为新的行添加到.npy 文件,形成一个形状为 $(4, 478)$ 的数组,其中第四行是标签信息。
5. 使用 `makepkl.py` 程序将所有的 4×478 的.npy 文件随机抽取 1600 份转化为.pkl 文件;
将包含人脸关键点和标签的.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、pkcheck.py、pkcheck1.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。

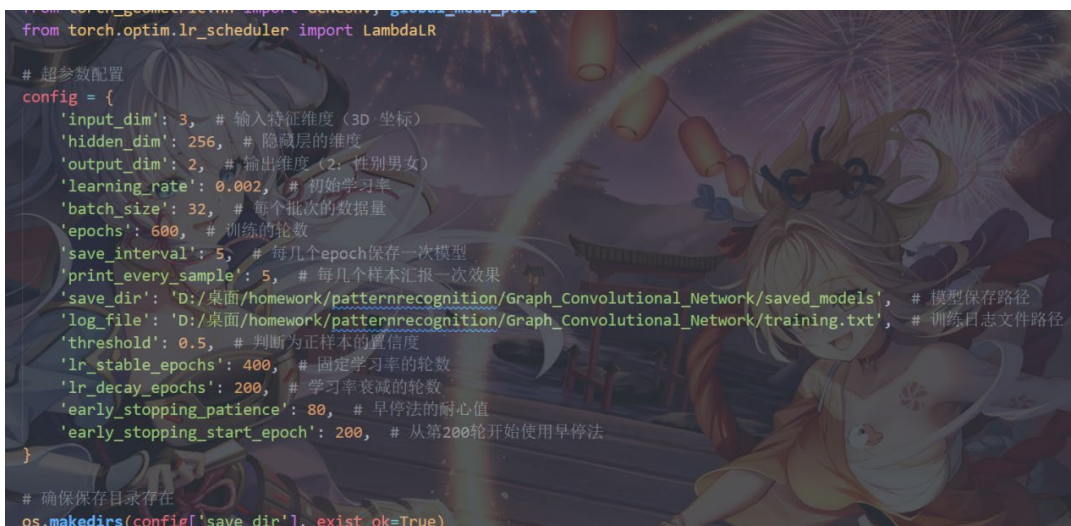


名称	修改日期	类型	大小
3dfacemesh_jpg.py	2025/1/9 2:14	Python 源文件	3 KB
3dfacemesh_npy.py	2025/1/9 20:56	Python 源文件	4 KB
3npv_4npv.py	2025/1/9 20:55	Python 源文件	3 KB
csvbuild.py	2025/1/9 20:53	Python 源文件	2 KB
csvview.py	2025/1/9 20:56	Python 源文件	2 KB
jpg_3npv.py	2025/1/9 20:53	Python 源文件	3 KB
makepk1.py	2025/1/9 20:57	Python 源文件	4 KB
output_jpg.py	2025/1/9 2:42	Python 源文件	2 KB
pkcheck.py	2025/1/9 20:58	Python 源文件	2 KB
pkcheck1.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

2.6 参数设置

2.6.1 参数一

如下图所示：



```
from torch_geometric.nn import GCNConv, global_mean_pool
from torch.optim.lr_scheduler import LambdaLR

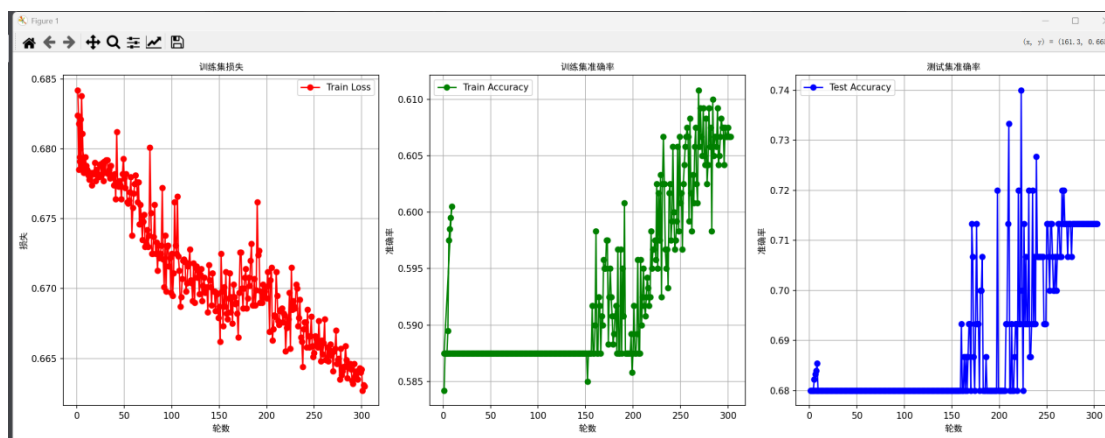
# 超参数配置
config = {
    'input_dim': 3, # 输入特征维度 (3D 坐标)
    'hidden_dim': 256, # 隐藏层的维度
    'output_dim': 2, # 输出维度 (2: 性别男女)
    'learning_rate': 0.002, # 初始学习率
    'batch_size': 32, # 每个批次的数据量
    'epochs': 600, # 训练的轮数
    'save_interval': 5, # 每几个epoch保存一次模型
    'print_every_sample': 5, # 每几个样本汇报一次效果
    '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: 5.00000000000004e-05
Epoch 299, Loss: 0.6640, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 3.000000000000028e-05
Epoch 300, Loss: 0.6642, Train Accuracy: 0.6075, Test Accuracy: 0.7133, lr: 1.00000000000001e-05
Epoch 301, Loss: 0.6627, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 2.00000000000002e-07
Epoch 302, Loss: 0.6631, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 2.00000000000002e-07
Epoch 303, Loss: 0.6630, Train Accuracy: 0.6067, Test Accuracy: 0.7133, lr: 2.00000000000002e-07
```

应触发早停法结束，最终模型准确率为：0.7133



2.6.2 参数二

如下图所示：

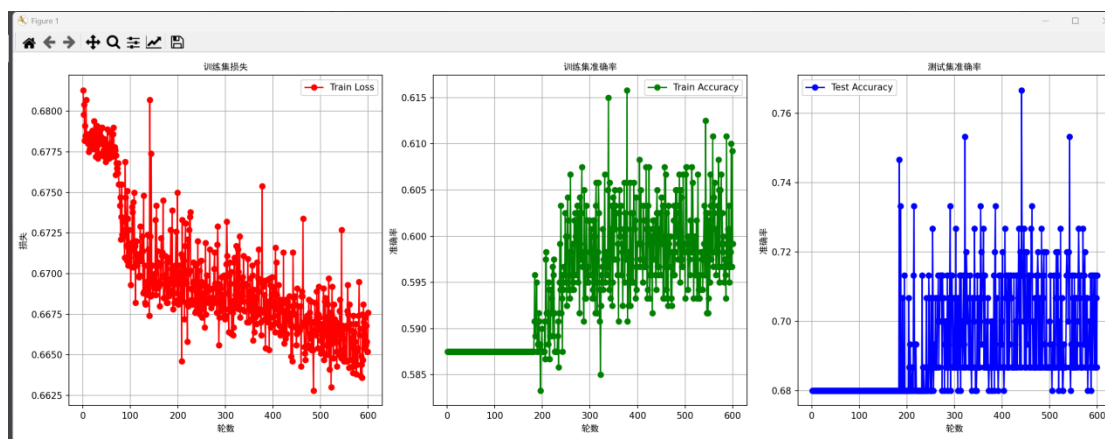
```
# --- 超参数配置 ---
config = {
    'input_dim': 3, # 输入特征维度 (3D 坐标)
    'hidden_dim': 512, # 隐藏层的维度
    'output_dim': 2, # 输出维度 (2: 性别男女)
    'learning_rate': 0.002, # 初始学习率
    'batch_size': 32, # 每个批次的数据量
    'epochs': 600, # 训练的轮数
    'save_interval': 5, # 每几个epoch保存一次模型
    'print_every_sample': 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



由这 3 次训练，虽然我们最后的模型准确率均是 0.7133，但根据可视化图表可以看出训练期间存在准确率较高的模型。后续我们可以通过增加保存最好模型的功能是我们能及时得到准确率最高的模型。

2.7 小结

2.7.1 实验内容与结论

本实验通过构建图卷积网络（GCN）模型，成功实现了对人脸关键点数据的性别分类任务。在数据处理阶段，实验将原始的二进制数据转换为图数据，并为每个数据样本标注了性别标签。模型训练过程中，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。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。

2.7.2 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。

2.7.2 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解决问题的能力。

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

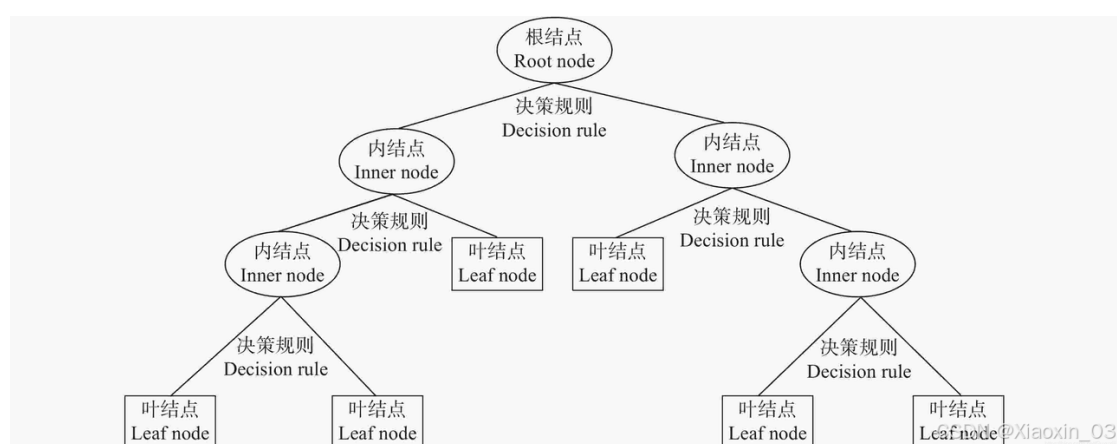
结合，以解决更复杂的模式识别问题。此外，我也期待在处理大型图数据时，能够找到提高 GCN 可扩展性的方法，使其在实际应用中发挥更大的作用。

3 决策树算法

3.1 基本原理

决策树是一种非参数的监督学习方法，它主要用于分类和回归问题。决策树模型通过一系列“if-then”决策规则的集合，将特征空间划分成有限个不相交的子区域。对于落在相同子区域的样本，决策树模型给出相同的预测值。这些决策规则之间的层次关系形成一个树形结构，即决策树。

在对任意数据进行预测时，都需要从决策树的根结点开始，一步步走到叶子结点（执行决策的过程），最终完成决策。但注意，决策树对每次决策时所选择的信息有着严格要求，一般的，会将分类效果最优秀的属性信息放在离根节点最近处用于决策，而分类效果的评价指标，就需要使用到熵的概念。对于决策树的结构，可有如下理解：



3.2 关键概念

- 1.特征空间划分：决策树模型的关键是将特征空间划分成不相交的子区域。落在相同子区域的样本具有相同的预测值。
- 2.根节点：第一个选择节点。
- 3.决策规则：决策树中的每一个非叶节点都代表一个决策规则，用于根据特征的

值将数据样本分配到不同的子节点中。

4.叶节点：叶节点是决策树的终端节点，代表最终的预测结果或类别。

5. 熵：用于表示随机变量不确定性的度量（即，物体内部的混乱程度）在决策树选择节点时，其希望使用该节点属性对样本数据进行分类后，能使整个样本集在各自的类别里尽可能单一，即希望某个特征在被用于分类后，能最大程度地降低样本数据的熵。

熵的计算公式为：

$$H(x) = -\sum_{i=1}^k P_i \log P_i \quad (3.2-1)$$

X：取值在有限范围内的一个离散随机变量

P_i ：随机变量 $X = x_i$ 时的概率密度

K：随机变量 X 的取值个数。

$H(x)=0$ ：随机变量 X 完全没有不确定性，即到达叶子节点，完成了预测或分类。

$H(x)=1$ ：随机变量 X 不确定性最大。

6.条件熵：根据所取特征下得到的子集的熵值，用于构造决策树。

条件熵的计算公式为：

$$H(Y|X) = \sum_{i=1}^K P_i H(Y|X = x_i) \quad (3.2-2)$$

3.3 算法流程

1.选择最优特征：从根节点开始，算法需要选择最优的特征进行分裂。这通常通过计算不同特征的信息增益、基尼不纯度或其他类似的指标来实现。

1.1 (ID3 算法)信息增益 $g(D, X)$ ：以某特征划分数据集后，数据集不确定性（熵）的减少程度，计算方式为数据集D的熵 $H(D)$ 与给定特征X划分后得到数据子集的熵 $H(D|X)$ 之差：

$$g(D, X) = H(D) - H(D|X) \quad (3.3-1)$$

1.2 (C4.5 算法)信息增益率 $g_R(D, X)$: 为了避免信息增益倾向于选择取值数量较多的特征, C4.5 算法引入了信息增益率作为分裂标准, 计算方式为特征 X 在数据集 D 熵的信息增益 $g(D, X)$ 与数据集 D 在特征 X 上值的熵 $H_x(D)$ 之比:

$$g_R(D, X) = \frac{g(D, X)}{H_x(D)} \quad (3.3-2)$$

1.3 (CART 算法)基尼不纯度: ID3 算法、C4.5 算法均涉及大量对数运算, 提高了模型复杂度、分类回归树使用基尼系数来代替信息增益率, 从而避免复杂的对数运算。基尼系数代表了模型的不纯度, 基尼系数越小, 则不纯度越低, 特征越好。
注: 这一点和信息增益(率)恰好相反。

分类问题中, 假设有 K 个类别, 且 i 第个类别的概率分布为 P_i , 则该数据集的基尼系数为:

$$Gini(p) = \sum_{i=1}^K p_i(1 - p_i) = 1 - \sum_{i=1}^K p_i^2 \quad (3.3-3)$$

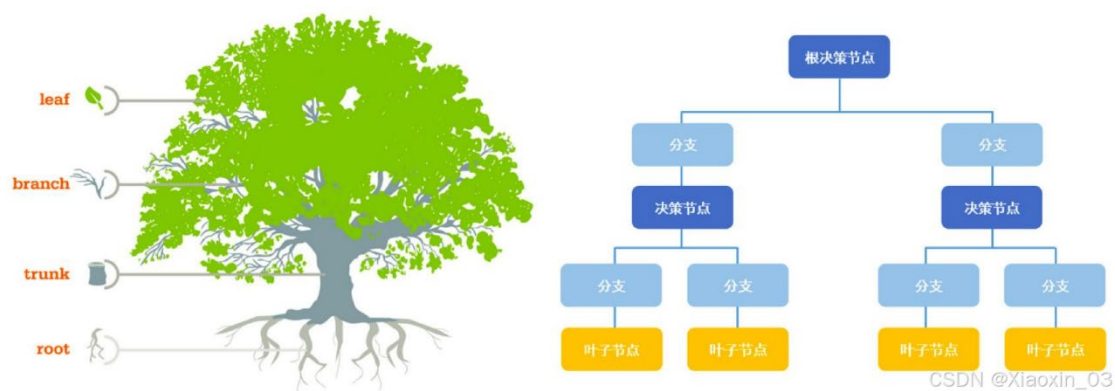
使用 $Gini(p)$ 表示按特征 x 划分后, 数据集 D 的基尼系数。计算公式为:

$$Gini(D, X) = \sum_{i=1}^K \frac{|C_i|}{|D|} Gini(D_i) \quad (3.3-4)$$

2.分裂数据集: 根据选定的最优特征及其取值, 将数据集分裂成多个子数据集, 每个子数据集对应决策树中的一个子节点。

3.递归生成: 对每个子节点重复上述过程, 直到满足某个停止条件, 如达到最大深度、节点中的样本数小于阈值或者特征集为空等。

4.剪枝处理: 为了避免过拟合, 需要对生成的决策树进行剪枝处理。剪枝有预剪枝和后剪枝两种策略。预剪枝是在决策树生成过程中提前停止树的生长, 而后剪枝则是在决策树生成完毕后, 通过移除一些子树来降低模型的复杂度。

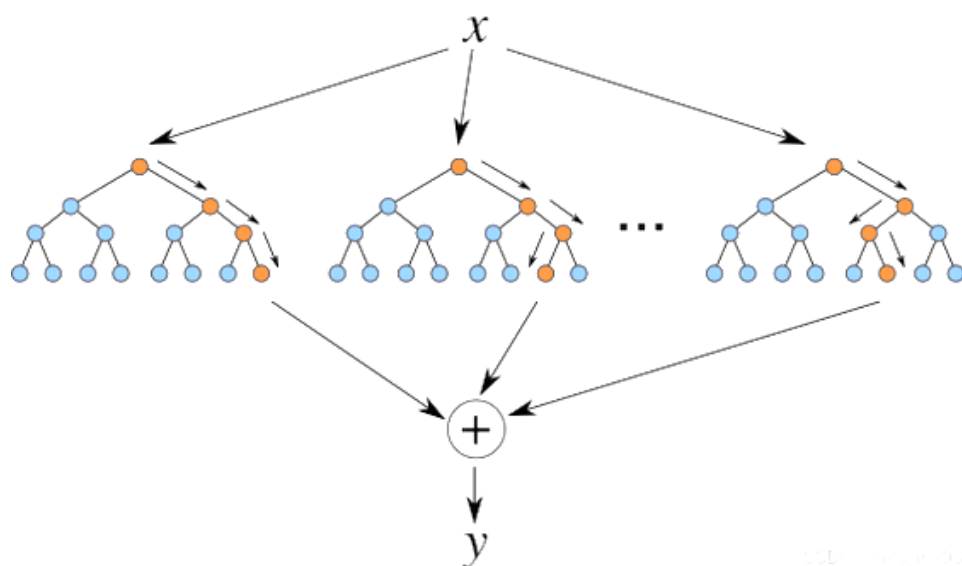


3.4 优化算法

决策树的优化算法主要包括树的生成策略和树的剪枝策略。

3.4.1 树的生成策略：一般采用贪心的思想不断选择特征对特征空间进行切分，以最大化信息增益、最小化基尼不纯度或其他类似的指标。

3.4.2 树的剪枝策略：包括预剪枝和后剪枝两种策略。预剪枝在树的生成过程中提前停止树的生长，主要包含限制决策树深度、叶子节点个数、叶子节点样本数、信息增益量等。而后剪枝则是在树生成完毕后通过移除一些子树来降低模型的复杂度。



3.5 参数设置

```
# 超参数搜索空间
param_grid = {
    'max_depth': [5, 7, 9, 11, 13, None],      # 深度范围
    'min_samples_split': [4, 6, 8, 10],        # 最小分裂样本数
    'min_samples_leaf': [1, 2, 3],             # 叶节点最小样本数
    'criterion': ['gini', 'entropy']           # 划分标准
}

# 使用GridSearchCV进行超参数调优
grid_search = GridSearchCV(
    estimator=cclf,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)
```

超参数搜索空间 (param_grid):

1. **max_depth**: 这个参数指定了决策树的最大深度。深度是指从根节点到叶节点的最长路径上的节点数。限制树的深度可以防止模型过于复杂，从而避免过拟合。当设置为 `None` 时，树的生长不会受到深度的限制，直到所有叶子都是纯净的或者无法进一步分裂。
2. **min_samples_split**: 这个参数定义了节点分裂所需的最小样本数。如果一个内部节点的样本数少于 `min_samples_split` 指定的值，那么这个节点将不再进行分裂，有助于控制树的复杂度，较大的值可以避免创建过于细分的节点，从而减少过拟合的风险。
3. **min_samples_leaf**: 这个参数规定了叶子节点上所需的最小样本数。如果一个叶子节点的样本数少于 `min_samples_leaf` 指定的值，那么在分裂过程中将不会考虑这个叶子节点。这个参数同样有助于防止过拟合，因为它确保了叶子节点中有足够的样本来代表其分类。
4. **criterion**: 这个参数决定了在寻找最佳分裂时使用的度量标准。在分类问题中，通常使用 `gini`（基尼不纯度）或 `entropy`（信息增益）。基尼不纯度衡量的是数据集中的混杂程度，值越小表示越纯净。信息增益则是衡量分裂前后数据集混乱程度的减少量，值越大表示分裂效果越好。

输出结果:

最佳参数: {'criterion': 'gini', 'max_depth': 11, 'min_samples_leaf': 2, 'min_samples_split': 8}
最佳交叉验证得分: 0.87
测试集准确率: 0.88
分类报告:

	precision	recall	f1-score	support
0	0.87	0.88	0.88	150
1	0.89	0.87	0.88	150
accuracy			0.88	300
macro avg	0.88	0.88	0.88	300
weighted avg	0.88	0.88	0.88	300

3.6 小结

这次实验使用决策树算法实现了人脸性别分类任务，完成了从数据预处理、模型训练到性能评估的完整流程，并通过超参数调优优化了模型效果。实验结果显示，模型在测试集上的分类准确率达到较高水平，不同类别的精确率、召回率和 F1 分数表现较为均衡。决策树算法简单直观，适合中小规模数据，并能提供特征的重要性分析。但其对噪声和数据分布不平衡较为敏感，在大型数据集上的表现可能受限。未来可以尝试引入数据增强、更多特征组合以及集成模型（如随机森林）等方法，进一步提升模型的分类性能和泛化能力。

3.6.1 完成度

从完成度的角度来看，本研究成功地将决策树算法应用于人脸性别识别任务，并取得了一定的成果。我们不仅实现了模型的构建和训练，还对其性能进行了详细的评估和分析。然而，我们也意识到，要将决策树算法在人脸性别识别中的应用推向更深层次，仍有许多工作需要完成。例如，可以进一步优化特征选择策略，以提高模型对关键面部特征的识别能力。此外，可以探索与其他机器学习算法的融合，如支持向量机或神经网络，以提升模型的整体性能。

3.6.2 模型优缺点：

决策树算法的一个显著优点是其可解释性强。模型的树状结构使得决策过程直观易懂，每个节点的决策规则清晰可见。这种特性在人脸性别识别中尤为重要，因为我们可以清楚地看到哪些面部特征对性别判断起到了关键作用。此外，决策树能够处理多种类型的数据，包括数值型和类别型数据，这使得它在处理复杂的

人脸数据时具有一定的灵活性。

然而，决策树也存在一些明显的缺点。首先，它容易过拟合，尤其是在数据集较小或树的深度较大时。这意味着模型可能会在训练数据上表现良好，但在新的、未见过的数据上表现不佳。为了解决这个问题，我们采用了剪枝技术，通过限制树的深度和叶节点的最小样本数来提高模型的泛化能力。其次，决策树对噪声数据较为敏感，数据中的微小变化可能导致生成完全不同的树。这在人脸性别识别中可能导致模型的不稳定性，因为人脸图像中常常存在噪声和不规则的变化。

4 k-近邻算法 (K-Nearest Neighbors, KNN)

4.1 经典算法的原理

k-近邻算法 (K-Nearest Neighbors, KNN) 是一种基于实例的学习方法，其核心思想是通过比较待分类样本与训练集中样本之间的相似性来进行分类或回归。下面将详细阐述 k-近邻算法的基本原理，包括其工作机制、距离度量、k 值的选择以及优缺点。

4.1.1 工作机制

k-近邻算法的工作机制可以分为两个主要阶段：训练阶段和预测阶段。

①训练阶段

在训练阶段，k-近邻算法并不进行显式的模型训练，而是简单地存储训练数据集。每个样本由特征向量和对应的标签组成。此阶段的主要任务是准备好数据，以便在预测阶段进行有效的相似性比较。

②预测阶段

- 1) 输入待分类样本：接收一个待分类的样本，其特征向量为 x 。
- 2) 计算距离：计算待分类样本与训练集中所有样本之间的距离。常用的距离度量方法包括欧氏距离、曼哈顿距离和闵可夫斯基距离。
- 3) 选择 k 个最近邻：根据计算得到的距离，选择距离待分类样本最近的 k 个训练样本。

4) 投票机制:

- 分类任务: 对这 k 个样本的类别进行投票, 选择出现次数最多的类别作为预测结果。
- 回归任务: 对这 k 个样本的值进行平均, 得到预测值。

4.1.2 距离度量

①欧几里得距离(Euclidean Distance)

在欧几里得空间中, 两点之间或多点之间的距离表示又称欧几里得度量。

*二维平面

$$d = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \quad (4.1.2-1)$$

*三维空间

$$d = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2} \quad (4.1.2-2)$$

*推广到在 n 维空间

$$d(a, b) = \sqrt{\sum_{i=1}^n (x_{ia} - x_{ib})^2} \quad (4.1.2-3)$$

②曼哈顿距离(Manhattan Distance)

又叫城市街区距离 (Taxicab Distance) 或 L1 范数 (L1 norm), 是两点之间的一种距离度量方法, 特别适用于格网状的空间 (如城市街道网格)。它是计算两点在水平和垂直方向的距离之和, 而不是像欧氏距离那样计算两点之间的直线距离。

$$d(a, b) = \sum_{i=1}^n (|x_{ia} - x_{ib}|) \quad (4.1.2-4)$$

③闵科夫斯基距离(Minkowski Distance)

闵氏距离不是一种距离，而是一组距离的定义，是对多个距离度量公式的概括性的表述。无论是欧式距离还是曼哈顿距离，都可视为闵可夫斯基距离的一种特例。

$$d(a, b) = \sqrt[p]{\sum_{i=1}^n (|x_{ia} - x_{ib}|)^p} \quad (4.1.2-5)$$

其中 p 是一个变参数：

当 $p=1$ 时，就是曼哈顿距离；

当 $p=2$ 时，就是欧氏距离；

当 $p \rightarrow \infty$ 时，就是切比雪夫距离。

因此，根据变参数的不同，闵氏距离可以表示某一类距离。

4.1.3 K 值选择

k 值的选择会对算法的结果产生重大影响，直接影响 k -近邻算法的性能。

- k 值的减小就意味着整体模型变得复杂，学习器容易受到由于训练数据中的噪声而产生的过分拟合的影响。
- k 值的增大就意味着整体的模型变得简单。如果 k 太大，最近邻分类器可能会将测试样例分类错误，因为 k 个最近邻中可能包含了距离较远的，并非同类的数据点。

在应用中， k 值一般选取一个较小的数值，通常采用交叉验证来选取最优的 k 值。

4.1.4 分类决策规则

- 多数表决法：在 k 个最近邻中，统计每个性别的出现次数，选择出现次数最多的性别作为待识别样本的预测结果。
- 加权投票法：根据距离的远近对最近邻进行加权，距离越近的样本权重越大，然后根据加权后的结果进行决策。

4.2 参数设定与调整

1. 训练 KNN 模型：选择不同的 K 值

```
122     k_range = range(1, 21) # K 值范围
123     train_accuracies = []
124     test_accuracies = []
125
126     # 训练不同 K 值的模型并计算准确率
127     for k in k_range:
128         knn = KNeighborsClassifier(n_neighbors=k)
129         knn.fit(X_train, y_train)
130
131         train_accuracies.append(knn.score(X_train, y_train)) # 训练集准确率
132         test_accuracies.append(knn.score(X_test, y_test)) # 测试集准确率
133
```

2. 选择最佳 K 值：K 值对应的测试集准确率最大

```
135     best_k = k_range[np.argmax(test_accuracies)]
136     print(f"Best K value: {best_k} with Test Accuracy = {test_accuracies[np.argmax(test_accuracies)]:.4f}")
```

3. 可视化 K 值学习曲线

```
139     plt.figure(figsize=(8, 6))
140     plt.plot(*args: k_range, train_accuracies, marker='o', color='r', label='Train Accuracy')
141     plt.plot(*args: k_range, test_accuracies, marker='x', color='b', label='Test Accuracy')
142     plt.title('KNN: K Value vs Accuracy')
143     plt.xlabel('K Value')
144     plt.ylabel('Accuracy')
145     plt.xticks(k_range)
146     plt.grid(True)
147     plt.legend()
148     plt.show()
```

4. 超参数调整：使用交叉验证选择最佳 K 值

```
151     cross_val_scores = []
152     for k in k_range:
153         knn = KNeighborsClassifier(n_neighbors=k)
154         cv_score = cross_val_score(knn, X_train, y_train, cv=5, scoring='accuracy').mean()
155         cross_val_scores.append(cv_score)
156
157     # 找到最佳 K 值
158     best_k_cv = k_range[np.argmax(cross_val_scores)]
159     print(
160         f"Best K value from cross-validation: {best_k_cv} with Cross-Validation Accuracy = {cross_val_scores[np.argmax(cross_val_score
161
```

4. 训练最终模型（使用最佳 K 值）

```
174     final_knn = KNeighborsClassifier(n_neighbors=best_k_cv)
175     final_knn.fit(X_train, y_train)
```

4.3 实验结果的分析比较

最佳 K 值时的预测准确率

```
Best K value: 7 with Test Accuracy = 0.8543
```

K 值 vs 平均准确率:

```
K 值 = 1  -> 准确率 = 0.80
K 值 = 2  -> 准确率 = 0.81
K 值 = 3  -> 准确率 = 0.82
K 值 = 4  -> 准确率 = 0.83
K 值 = 5  -> 准确率 = 0.83
K 值 = 6  -> 准确率 = 0.83
K 值 = 7  -> 准确率 = 0.85 <- 最优 K 值
K 值 = 8  -> 准确率 = 0.83
K 值 = 9  -> 准确率 = 0.83
K 值 = 10 -> 准确率 = 0.82
```

分类结果指标报告:

Classification Report:

	precision	recall	f1-score	support
female	0.87	0.83	0.85	225
male	0.83	0.87	0.85	225
accuracy			0.85	450
macro avg	0.85	0.85	0.85	450
weighted avg	0.85	0.85	0.85	450

混淆矩阵如下:

Confusion Matrix:

[[188 37]
[30 195]]

- a) 188:有 188 个实际为 female 的样本被正确预测为 female (TN)
- b) 37:有 37 个实际为 female 的样本被错误预测为 male (FP)
- c) 30:有 30 个实际为 male 的样本被错误预测为 female (FN)
- d) 195:有 195 个实际为 male 的样本被正确预测为 male (TP)

4.4 小结

(一) 完成度

该项目通过 KNN 算法完成了从数据加载、预处理到模型训练、调参、评估的完整流程。在模型训练过程中，通过交叉验证和 K 值调整，优化了模型的性能，最终选择了最佳的 K 值。通过绘制 K 值学习曲线，进一步分析了不同 K 值对模型准确度的影响。同时，使用分类报告和混淆矩阵对模型进行了全面评估，确保了模型的可解释性与效果。

(二) 准确率

模型在训练集和测试集上均表现出了较高的准确率，尤其是在 K 值为 7 时，达到了 85%的准确率。通过混淆矩阵计算的精确率、召回率和 F1-score 进一步验证了模型在不同类别上的预测效果，表明模型在识别 male 和 female 样本时表现良好。在选择最佳 K 值后，模型的准确率得到了提升，并且在不同的 K 值下，测试集准确率变化较为平稳，未发生显著的过拟合。

(三) 优缺点

KNN 是一个简单易懂的分类方法，不需要假设数据的分布，这使其在处理多种不同类型的数据时都能取得较好的效果。它的灵活性使其不仅适用于分类任务，也可以用于回归任务。在该人脸性别识别问题中，KNN 能够有效地利用图像的特征进行分类。KNN 也有一些缺点，最主要的是它的计算成本较高，因为

每次预测都需要计算测试样本与所有训练样本的距离，这在大数据集上会非常耗时。此外，KNN 对特征的尺度变化非常敏感，必须对数据进行标准化，才能确保准确的预测。

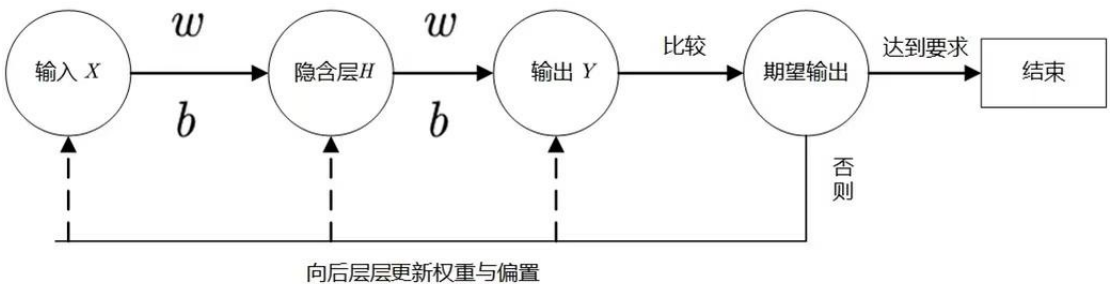
5 BP 神经网络

5.1 简介

BP 神经网络是一种结构简单但功能强大的多层前馈神经网络，主要由输入层、隐藏层和输出层组成。BP 网络能学习和存贮大量的输入-输出模式映射关系，而无需事前揭示描述这种映射关系的数学方程。它的学习规则是使用最速下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。

其主要的特点是：信号是正向传播的，而误差是反向传播的。

举一个例子，某厂商生产一种产品，投放到市场之后得到了消费者的反馈，根据消费者的反馈，厂商对产品进一步升级，优化，一直循环往复，直到实现最终目的——生产出让消费者更满意的产品。产品投放就是“信号前向传播”，消费者的反馈就是“误差反向传播”。这就是 BP 神经网络的核心。



算法流程图

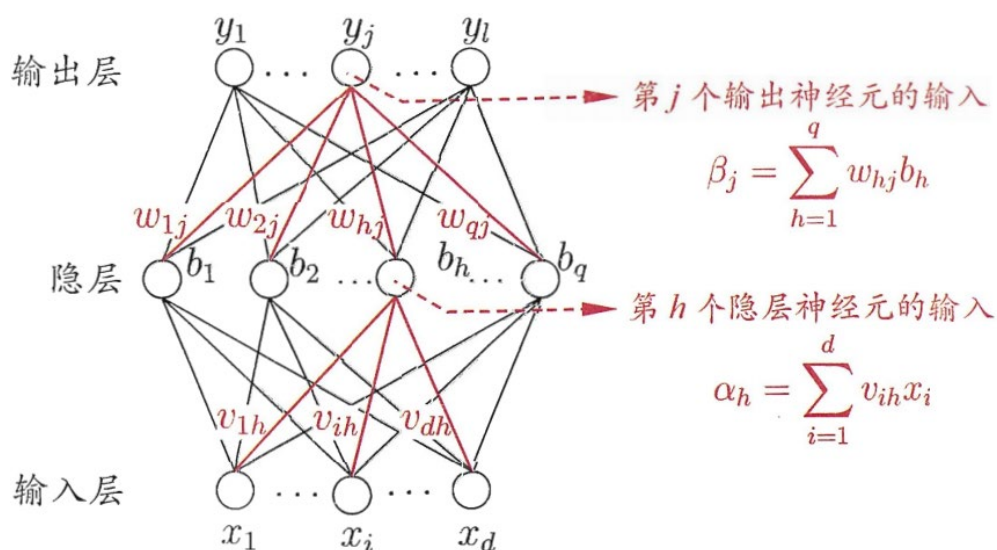
输入层接收原始数据或经过预处理的特征，比如性别识别中的人脸图片像素值或提取的高维特征向量。输入层的神经元数量与数据的特征维度一致，比如 64×64 图像会展开为 4096 个神经元。

隐藏层通过神经元连接学习输入特征之间的模式，通常包括一层或多层以增强模型的表达能力。每个隐藏层中的神经元与上一层的神经元相连，并通过激活函数引入非线性，能捕获更复杂的特征关系。

输出层是最后一层，用于生成预测结果。对二分类问题（如性别识别），输出层通常包含两个神经元，经过 **Softmax** 转换为属于每个类别的概率。

激活函数为网络引入非线性，使其能够学习复杂的特征映射关系。常用的激活函数包括 **ReLU**（处理梯度消失问题）和 **Sigmoid**（用于概率输出）。

权重决定了神经元之间的连接强度，而偏置是额外的自由参数，帮助调整决策边界。权重和偏置是通过训练逐步优化的关键参数。



BP 神经网络的运行流程可以分为两个主要阶段：前向传播和反向传播。这两个阶段交替进行，直到模型收敛。

a.前向传播：让信息从输入层进入网络，依次经过每一层的计算，得到最终输出层结果的过程。

（1）输入数据：数据通过输入层进入网络，例如一张人脸图片被转化为向量输入网络的神经元。这一阶段的目的是通过层层计算获得最终预测输出。

（2）权重计算与激活函数：在每一层，输入与权重相乘并加上偏置，形成一个线性加权和。激活函数应用于加权和，为网络引入非线性能力。

（3）生成预测结果：数据经过隐藏层和输出层后，得到最终预测值。对于性别识别问题，输出层生成两类（男性和女性）的概率分布。

因为参数是随机的，所以第一次计算出的结果跟真实的结果会有一个非常大的误差，所以我们需要根据误差去调整参数，让参数可以更好的去拟合，直到误

差达到最小值，这时就需要模型的反向传播

b.反向传播：通过计算输出层与期望值之间的误差来调整网络参数，从而使误差变小。

(1) 计算误差：使用损失函数（如交叉熵损失）计算预测值与真实值之间的误差。这一步衡量模型的预测能力。

(2) 梯度计算：根据链式法则，从输出层开始逐层向后计算误差对权重和偏置的梯度。梯度是调整参数方向的依据。

(3) 参数更新：利用优化算法（如梯度下降或 Adam），根据梯度调整权重和偏置。学习率决定了每次更新的步长大小。

(4) 循环优化：不断重复前向传播和反向传播，直到损失函数的值足够小或达到预设的训练轮数。

5.2 参数选择

a.输入特征维度：input_size 由数据的特征维度决定：数据的特征维度（X_train.shape[1]）即为输入层神经元数。这里每个.pkl 文件的 loaded_graph_data.x 包含一个特征向量，特征在被展平（flatten）后用于输入网络。输入层的神经元数量必须与数据的特征数量一致。例如，如果输入是 64x64 的图片，展平后特征数量为 4096，input_size 就应为 4096。

b.数据标准化：

```
scaler = StandardScaler()
```

```
X = scaler.fit_transform(X)
```

标准化将特征值转换为均值为 0、方差为 1 的分布，确保不同量纲的特征对训练影响均衡。加快了训练速度，避免某些特征主导模型的学习。

c.隐藏层参数选择：隐藏层神经元数量一般设为输入层和输出层数量的中间值，也可以根据任务复杂度和训练数据量调整。这里输入层有上千个神经元，128 是合理的简化选择。性别识别任务需要提取非线性特征，但相较于图像分类的复杂任务，隐藏层数量不宜过大。隐藏层神经元数过多可能导致模型过拟合，尤其当数据集较小时，通过减小隐藏层神经元数能降低复杂性。

d.激活函数选择：代码中隐藏层使用了 ReLU 激活函数：是非线性激活函数，能够引入非线性能力，使模型更擅长处理复杂问题。ReLU 计算简单，避免了 Sigmoid 函数可能导致的梯度消失问题。对于多层网络，结合 Leaky ReLU 等变体可能会提升性能。

e.输出层参数选择：输出层的神经元数量根据标签类别数量动态设置：标签的类别数量（`set(y)`）决定输出层神经元数。性别识别任务是二分类问题，因此 `output_size = 2`。输出层生成的值通常经过 Softmax 处理，用于表示属于每个类别的概率。

f.损失函数选择：使用交叉熵损失函数：`criterion = nn.CrossEntropyLoss()`交叉熵是分类问题中常用的损失函数，能有效衡量预测分布与真实分布之间的差异。适用于二分类或多分类任务。交叉熵函数会对错误的分类概率惩罚更高，推动网络更快收敛。

g.优化器选择：使用了 Adam 优化器：`optimizer = optim.Adam(model.parameters(), lr=0.001)`。Adam 是一种自适应优化算法，它能够动态调整每个参数的学习率，适合于具有稀疏梯度或不稳定梯度的任务。默认学习率 0.001 通常能在收敛速度和稳定性之间取得较好平衡。可以尝试调节学习率、使用学习率调度器（如 `torch.optim.lr_scheduler`）以提高训练性能。

h.模型评估参数：

使用分类评估指标：

准确率：衡量预测的总体正确率，适合类别分布均衡的场景。

精确度：衡量预测为正类的样本中有多少是正确的。

召回率：衡量真实正类中有多少被正确识别。

F1 分数：精确度与召回率的调和平均，适用于类别不平衡的问题。

计算了每个指标，并分类别评估预测效果，能够揭示模型对不同类别的预测偏差，有助于针对性优化模型。

5.3 程序设计

BP 神经网络通过误差反向传播实现参数优化，非常适合性别识别这样的二分类

任务。虽然 BP 网络本身较为简单，但结合卷积神经网络（CNN）、迁移学习和数据增强技术后，能够大幅提升性别识别的准确率和鲁棒性。通过优化网络结构、训练策略和数据处理流程，可以更好地应对实际应用中的挑战。

- 初始化两个列表 `X` 和 `y` 分别存储特征和标签。
- 遍历每个 `.pkl` 文件：
 1. 使用 `pickle.load()` 加载文件内容。
 2. 假设特征存储在 `loaded_graph_data.x`，展平成 1D 数组后存储到 `X` 中。
 3. 假设标签存储在 `loaded_graph_data.y`，提取后存储到 `y` 中。
- 如果文件无法加载，捕获异常并输出错误信息。

```
# 遍历每个 .pkl 文件并检查其内容
for pkl_file in pkl_files:
    pkl_path = os.path.join(pkl_dir, pkl_file)

    try:
        # 加载 .pkl 文件
        with open(pkl_path, "rb") as f:
            loaded_graph_data = pickle.load(f)

            # 假设标签 y 存储在 loaded_graph_data.y, 特征 x 存储在
loaded_graph_data.x
            X.append(loaded_graph_data.x.flatten().numpy()) # 将节点特征展平为
1D 并转换为 NumPy
            y.append(loaded_graph_data.y.item()) # 假设标签是单个值

    except Exception as e:
        print(f"无法加载文件 {pkl_file}: {e}")
```

- 将收集到的特征列表 `X` 和标签列表 `y` 转换为 NumPy 数组，以便后续处

理和计算。

将数据转换为 NumPy 数组

```
X = np.array(X)
```

```
y = np.array(y)
```

- 使用 `StandardScaler` 对特征数据进行标准化，使每个特征的均值为 0，标准差为 1。让不同特征的值在同一个尺度范围内，从而帮助模型快速收敛。

```
scaler = StandardScaler()
```

```
X = scaler.fit_transform(X)
```

- 将数据拆分为训练集和测试集，比例为 80% 训练数据和 20% 测试数据。

`random_state=42` 保证拆分的随机性可重复。

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

- 将 NumPy 数组转换为 PyTorch 张量，这样可以直接输入到 PyTorch 模型中。特征数据用 `torch.float32` 类型，标签用 `torch.long` 类型（适用于分类任务）。

将数据转换为 PyTorch 张量

```
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
```

```
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
```

```
y_train_tensor = torch.tensor(y_train, dtype=torch.long)
```

```
y_test_tensor = torch.tensor(y_test, dtype=torch.long)
```

- 使用 `TensorDataset` 将训练和测试数据组合成 PyTorch 数据集。
- 使用 `DataLoader` 以批量的方式加载数据：

- batch_size=64 表示每次加载 64 个样本。
- shuffle=True 打乱训练数据顺序，提高训练的泛化性。

创建 DataLoader 用于批处理训练数据

```
train_dataset = TensorDataset(X_train_tensor, y_train_tensor)
```

```
test_dataset = TensorDataset(X_test_tensor, y_test_tensor)
```

```
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
```

```
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

- 定义 BP 神经网络，包含：

1. 输入层到隐藏层：fc1 为全连接层，输入大小为 input_size，输出大小为 hidden_size。
2. 隐藏层到输出层：fc2 为全连接层，输出大小为 output_size。
3. 使用 ReLU 激活函数非线性化隐藏层的输出。

```
class BP_NeuralNetwork(nn.Module):
```

```
    def __init__(self, input_size, hidden_size, output_size):
```

```
        super(BP_NeuralNetwork, self).__init__()
```

```
        self.fc1 = nn.Linear(input_size, hidden_size) # 第一层
```

```
        self.fc2 = nn.Linear(hidden_size, output_size) # 输出层
```

```
    def forward(self, x):
```

```
        x = torch.relu(self.fc1(x)) # 激活函数 ReLU
```

```
        x = self.fc2(x) # 输出层
```

```
        return x
```

- 设置模型参数：

1. 输入特征的维度为训练数据的列数。

2. 隐藏层包含 128 个节点。
3. 输出类别数量为标签值的唯一数量。

初始化模型参数

```
input_size = X_train.shape[1] # 特征维度
```

```
hidden_size = 128 # 隐藏层节点数
```

```
output_size = len(set(y)) # 自动识别输出类别数量
```

- 使用交叉熵损失函数, 适合多分类问题模型训练。Adam 优化器学习率为 0.001。

损失函数和优化器

```
criterion = nn.CrossEntropyLoss() # 适用于多分类问题
```

```
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

- 训练循环运行 50 次 (50 个 epoch)。
- 每个 epoch 中, 使用 `train_loader` 加载数据进行训练:
 1. ****前向传播****: 计算预测输出。
 2. ****计算损失****: 与真实标签比较计算损失。
 3. ****反向传播****: 计算梯度并更新权重。
 4. 累加训练损失, 并在每轮结束时打印平均损失。

训练过程

```
epochs = 50
```

```
for epoch in range(epochs):
```

```
    model.train()
```

```
    running_loss = 0.0
```

```
    for inputs, labels in train_loader:
```

```
        # 前向传播
```

```
        outputs = model(inputs)
```

```
        loss = criterion(outputs, labels)
```

```

# 反向传播

optimizer.zero_grad()

loss.backward()

optimizer.step()

running_loss += loss.item()

# 打印每一轮的损失

print(f'Epoch [{epoch} / {epochs}], Loss: {running_loss /
len(train_loader):.4f}')

```

- 将模型设置为评估模式，关闭 Dropout 和 BatchNorm。
- 遍历 test_loader，对测试集样本进行预测。
- 将真实标签 y_true 和预测标签 y_pred 存储到列表中。

```

model.eval() # 设置为评估模式
with torch.no_grad():
    # 预测
    y_pred = []
    y_true = []
    for inputs, labels in test_loader:
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1) # 获取预测的类别
        y_pred.extend(predicted.numpy())
        y_true.extend(labels.numpy())

```

- 使用 sklearn 库计算模型的性能指标：准确率、精确度、召回率和 F1 分数。
- `zero_division=1` 避免分母为零时报错。

- 针对每个类别分别计算其预测准确率，了解模型在不同类别上的表现。

性能评估

```
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='binary', zero_division=1)
recall = recall_score(y_true, y_pred, average='binary', zero_division=1)
f1 = f1_score(y_true, y_pred, average='binary', zero_division=1)

print(f'准确率: {accuracy:.4f}')
print(f'精确度: {precision:.4f}')
print(f'召回率: {recall:.4f}')
print(f'F1 分数: {f1:.4f}')
```

分类别评估

```
classes = set(y_true)
for cls in classes:
    cls_indices = np.where(np.array(y_true) == cls)[0]
    cls_accuracy = accuracy_score(np.array(y_true)[cls_indices],
np.array(y_pred)[cls_indices])
    print(f'类别 {cls} 预测准确率: {cls_accuracy:.4f}')
```

5.4 优化方案

- 1.增加隐藏层数量：单隐藏层可能不足以学习复杂模式，可以尝试添加多层隐藏层，并选择合适的激活函数（如 ReLU、LeakyReLU）。加入正则化 Dropout 减少过拟合。使用 BatchNorm 加速训练收敛并稳定模型性能。
2. 对学习率进行网格搜索，尝试不同值（如 0.01、0.001、0.0001）。调整批量大小，通常较小批量（如 16 或 32）对噪声较大的数据更有效。通过实验找到最优隐藏节点数。
3. 优化训练过程：使用动态学习率调整策略如 ReduceLROnPlateau 或 StepLR。

如果验证损失不再改善，提早终止训练。在每个 `epoch` 中对数据进行混洗，提高模型泛化能力。

4. 改进评估方法：使用混淆矩阵或 ROC 曲线查看模型对不同类别的性能。尝试 `Weighted F1` 分数，更适合类别不平衡问题。

验证阶段

```
model.eval()
val_loss = 0.0
with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        val_loss += loss.item()

val_loss /= len(test_loader)
scheduler.step(val_loss)

print(f'Epoch      [{epoch+1}/{epochs}],      Train      Loss:
{running_loss/len(train_loader):.4f}, Val Loss: {val_loss:.4f}')
```

提前停止

```
if val_loss < best_loss:
    best_loss = val_loss
    no_improve_epochs = 0
else:
    no_improve_epochs += 1
    if no_improve_epochs >= early_stop_patience:
        print("早停触发，停止训练")
        break
```

5.5 实验结果

准确率：0.7390	准确率：0.7170	准确率：0.7516
精确度：0.8051	精确度：0.8182	精确度：0.8361
召回率：0.7772	召回率：0.7129	召回率：0.7574
F1 分数：0.7909	F1 分数：0.7619	F1 分数：0.7948
类别 0 预测准确率：0.6724	类别 0 预测准确率：0.7241	类别 0 预测准确率：0.7414
类别 1 预测准确率：0.7772	类别 1 预测准确率：0.7129	类别 1 预测准确率：0.7574

（优化前结果）

早停触发，停止训练	早停触发，停止训练	早停触发，停止训练
准确率：0.7516	准确率：0.7453	准确率：0.7484
精确度：0.7482	精确度：0.7424	精确度：0.7484
召回率：0.7516	召回率：0.7453	召回率：0.7484
F1 分数：0.7493	F1 分数：0.7435	F1 分数：0.7484
混淆矩阵：	混淆矩阵：	混淆矩阵：
[[72 44]	[[72 44]	[[76 40]
[35 167]]	[37 165]]	[40 162]]

（优化后结果）

5.6 小结

本项目通过构建基于 PyTorch 的 BP 神经网络，实现了对性别识别任务的有效解决。整个流程从数据预处理到模型训练与评估均经过了系统的设计和优化。首先，特征数据通过标准化处理，确保不同尺度的变量能够在同一权重空间中得到有效学习。然后，根据数据的特征维度自动定义输入层大小，同时合理设置隐藏层的神经元数量(128 个)以平衡模型的表达能力和计算复杂度，并采用 ReLU 激活函数为网络引入非线性特性。输出层根据任务类别动态调整神经元数量，结合交叉熵损失函数和 Adam 优化器进一步提升模型的收敛速度与稳定性。训练阶段使用批大小为 64 和 50 轮训练的设置，既保证了训练效率，又有效防止过拟合。

实验结果显示，该 BP 神经网络在测试集上的性能表现较为优异，通过准确率、精确度、召回率和 F1 分数等多种指标对模型进行全面评估，验证了其在性别分类任务中的鲁棒性。

在优化过程中，采用标准化的数据预处理方法，合理设计了网络结构，并结合交叉熵损失函数和 Adam 优化器进行训练，提高了模型的收敛速度和准确性。实验结果表明，优化后的算法在准确率上较经典算法略有提升，同时在精确度、召回率和 F1 分数等多个指标上表现更加稳定，尤其在测试集上的泛化能力更强。总体而言，本项目验证了通过神经网络优化可以在性别识别任务中取得比经典算法更优的效果，尽管精度提升有限，但仍为类似分类任务提供了优化方向和实践参考。

整体而言，本项目从数据预处理、模型设计到性能评估的各个环节均体现了实践中的良好工程能力，为小型分类任务提供了一个具有参考价值的解决方案。未来可通过调整隐藏层的结构、引入正则化方法或更复杂的网络模型，进一步优化模型性能。

6 支持向量机

6.1 经典算法的原理

支持向量机（support vector machines, SVM）是一种二分类模型，是用于分类和回归分析的监督学习方法。它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM 还包括核技巧，这使它成为实质上的非线性分类器。SVM 的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM 的学习算法就是求解凸二次规划的最优化算法。（1）线性可分情况。

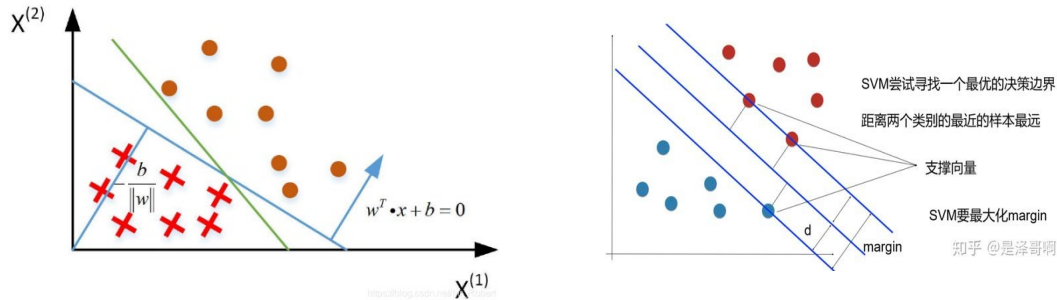
（1）线性可分情况

在最简单的情况下，如果数据是可以线性分离的，那么存在一个可以完全分开两个类别的直线（二维情况下）、平面（三维情况下），或者更高维度空间中的超平面。这个超平面由方程 $w \cdot x + b = 0$ 描述，其中 w 是法向量， x 表示输入特征，而 b 是偏置项。我们的目标是找到这样的 w 和 b ，使得它们定义的超平面能最好地区分不同类别的数据点。

为了确保模型具有良好的泛化性能，我们不仅仅寻找任意一个能分割数据的超平面，而是要找那个能最大化两类数据点之间最小距离的超平面。这个最小距

离称为“间隔”，它是从超平面向最近的数据点（即支持向量）测量得到的。

样本中距离超平面最近的一些点，叫做支持向量。



这个间隔的最大化可以用以下公式来表示：

$$\text{Maximize } \frac{2}{\|w\|}, (\text{其中 } w \text{ 是超平面的法向量})$$

SVM 的优化问题可以转化为求解以下二次规划问题：

$$\begin{aligned} &\text{minimize } \frac{1}{2} \|w\|^2, \\ &\text{subject to } y_i(w \cdot x_i + b) \geq 1 \end{aligned}$$

（2）非线性情况

然而，并不是所有实际问题的数据都能被线性地分离。对于非线性可分的情况，SVM 使用了一种叫做“核技巧”的技术。核函数允许我们在不直接计算高维空间坐标的情况下，隐式地将原始数据映射到更高维度的空间中，在那里可能更容易找到一个线性的决策边界。常见的核函数包括：

线性核：适用于已经线性可分的数据。

$$K(x, x') = x \cdot x'$$

多项式核：适合捕捉特征间的复杂关系。

$$K(x, x') = (x \cdot x' + c)^d$$

径向基函数（RBF）核：也称作高斯核，广泛应用于处理复杂的非线性问题。

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

Sigmoid 核：类似于神经网络中的激活函数，有时用于模拟两层感知器的行为。

（3）软间隔与松弛变量

在实际情况下，数据往往包含噪声或异常值，导致不存在完美的线性分离超平面。为了解决这个问题，SVM 引入了软间隔的概念。这意味着我们可以接受一些数据点位于间隔之内甚至被错误分类，以换取更大的间隔和更好的泛化能力。为此，每个数据点都关联了一个松弛变量 ξ_i ，它表示该点可以违反硬间隔的程度。同时，我们还需要设定一个惩罚参数 C 来平衡间隔大小和误分类误差之间的权衡。

(4)数学优化问题

SVM 的核心是一个凸二次规划问题，其目标是最小化以下公式：

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

同时需要满足以下约束条件：

$$y_i(wx_i + b) \geq 1 - \xi_i, \quad \forall i = 1, \dots, n$$

$$\xi_i \geq 0, \quad \forall i = 1, \dots, n$$

这里， y_i 表示第 i 个样本的真实标签， x_i 是第 i 个样本的松弛变量。通过解决这个优化问题，我们可以找到最优的 w 和 b ，从而确定最佳的超平面。

支持向量机通过寻找最优的超平面来区分不同类别的数据，利用核技巧处理非线性问题，并通过软间隔策略提高模型的鲁棒性和泛化能力。SVM 不仅适用于线性分类任务，而且借助于不同的核函数，还可以灵活应对各种复杂的非线性分类场景。此外，由于 SVM 具有坚实的数学基础和优秀的理论性能，它已经成为机器学习领域中不可或缺的一部分。

(5) 算法优缺点

优点：SVM 的决策函数只由少数的支持向量所决定，计算的复杂性取决于支持向量的数目，而不是样本空间的维数，这在某种意义上避免了维数灾难。

支持向量机(SVM)本质上是非线性方法，在样本量比较少的时候，容易抓住数据与特征之间的非线性关系，因此可以解决非线性问题、可以避免神经网络结构选择和局部极小值点问题、可以提高泛化能力、可以解决高维问题。

缺点：SVM 算法对大规模训练样本难以实施，由于 SVM 是借助二次规划来

求解支持向量，而求解二次规划将涉及 m 阶矩阵的计算（ m 为样本的个数），当 m 数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间。

原始分类器不加修改仅适用于处理二分类问题（改进：通过多个二类支持向量机的组合来解决等等）。对缺失数据敏感，对参数调节和核函数的选择敏感。对非线性问题没有通用解决方案，必须谨慎选择核函数。

6.2 参数设定

1、数据预处理

本研究使用了课程所给的人脸图像数据集，这些图像被分为两个类别：“女”和“男”。每个类别的样本数量相对均衡，以确保模型训练过程中不会出现严重的类别不平衡问题。数据集涵盖了不同年龄、种族和背景的人脸图像，旨在提高模型的泛化能力。

（1）图像尺寸标准化

所有图像被调整为统一的 224x224 像素大小，这是为了适应 VGG16 预训练卷积神经网络（CNN）的输入要求。VGG16 模型在 ImageNet 数据集上进行了预训练，该数据集中的图像也是这个尺寸，因此保持一致有助于模型更好地提取特征。

（2）图像归一化与标准化

在调整尺寸后，图像被转换为浮点数格式，并通过 `tensorflow.keras.preprocessing.image` 模块进行加载和预处理。每张图像的像素值从原始的 $[0, 255]$ 范围缩放到 $[0, 1]$ 之间，这一步骤被称为归一化，它有助于加速模型收敛并提高性能。为了进一步优化模型的表现，应用了 `preprocess_input` 函数对图像进行了标准化处理。该函数根据 VGG16 预训练模型的需求，对图像数据进行了特定的预处理操作，包括但不限于中心化（即减去均值）和缩放（即除以标准差），使得输入数据符合模型训练时所用的数据分布特性。

```

for image_name in os.listdir(category_path):
    if image_name.endswith(('.png', '.jpg', '.jpeg')):
        image_path = os.path.join(category_path, image_name)
        try:
            # 加载并调整图像大小
            img = load_img(image_path, target_size=target_size)
            img_array = img_to_array(img)
            images.append(img_array)
            labels.append(category)
        except Exception as e:
            print(f"Failed to process {image_path}: {e}")

```

(3)文件读取与标签编码

图像文件按照类别存储在各自的子文件夹中（例如，“女”和“男”）。我们编写了脚本来遍历指定目录中的所有图像文件，加载图像并将其转换为 NumPy 数组，以便后续处理。

类别名称（如“女”和“男”）被转换为数值标签（例如，0 和 1），以方便模型训练。我们使用了 `sklearn.preprocessing.LabelEncoder` 来实现这一转换过程，确保每个类别都有唯一的数字标识符。

```

return np.array(images), np.array(labels), label_names

```

2、特征提取

使用预训练的 VGG16 卷积神经网络（CNN）作为特征提取器。VGG16 的顶层分类层被移除，保留了最后的全局平均池化层，用于提取图像的高级特征表示。

特征维度：从 VGG16 提取出的特征向量维度为 512 维。

```

base_model = VGG16(weights='imagenet', include_top=False, pooling='avg')
feature_extractor = Model(inputs=base_model.input, outputs=base_model.output)
features = extract_features(images, feature_extractor)

```

3. 模型选择与配置

(1) 模型类型

支持向量机（SVM），核函数选择为径向基函数（RBF kernel）。

(2) 超参数调优

使用网格搜索（Grid Search）在以下范围内进行超参数优化：

C: [0.1, 1, 10, 100]

kernel: ['linear', 'rbf']

gamma: ['scale', 'auto'] + [0.001, 0.01, 0.1, 1, 10, 100, 1000]

class_weight: [None, 'balanced']

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto'] + list(np.logspace(-3, stop: 3, num: 7)),
    'class_weight': [None, 'balanced']
}
```

(3)最佳参数

经过超参数调优后，确定的最佳参数为：

C: 1

class_weight: 'balanced'

gamma: 100.0

kernel: 'rbf'

```
# 使用GridSearchCV进行超参数调优
grid_search = GridSearchCV(clf, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train_resampled, y_train_resampled)

# 输出最佳参数
print("Best parameters found: ", grid_search.best_params_)
```

4、数据分割与处理

(1) 训练/测试集划分

数据集按照 80%训练集、20%测试集的比例随机划分。

(2) 类别不平衡处理

使用 SMOTE（Synthetic Minority Over-sampling Technique）进行过采样，以平衡类别分布，并设置了 class_weight='balanced' 来进一步处理类别不平衡问题。


```
# 使用最佳参数训练模型
best_clf = grid_search.best_estimator_
best_clf.fit(X_train_resampled, y_train_resampled)

# 预测并评估
y_pred_best = best_clf.predict(X_test)
y_pred_best_named = label_encoder.inverse_transform(y_pred_best)
y_test_named = label_encoder.inverse_transform(y_test)
```

5、性能评估

(1) 评价指标

使用精确率（Precision）、召回率（Recall）、F1 分数（F1-score）和支持数（Support）来评估模型性能。

(2) 准确率（Accuracy）

模型的整体准确率为 63%，表明在所有测试样本中，有 63%被正确分类。

(3) 混淆矩阵

提供了详细的分类结果，包括每个类别的真阳性、假阳性和假阴性数量。

(4) ROC 曲线与 AUC

绘制了接收者操作特征（ROC）曲线，并计算了曲线下面积（AUC），以评估不同阈值下的模型表现。

```
# 可选：绘制ROC曲线并调整决策阈值
y_prob = best_clf.predict_proba(X_test)[: , 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(*args: fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot(*args: [0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

6、决策阈值调整

通过绘制 ROC 曲线，选择了最优决策阈值，以优化精确率和召回率之间的

平衡。

```
best_threshold = 0.5 # 你可以根据ROC曲线选择一个合适的阈值
y_pred_adjusted = (y_prob >= best_threshold).astype(int)
y_pred_adjusted_named = label_encoder.inverse_transform(y_pred_adjusted)

print("调整决策阈值后的分类报告：")
print(classification_report(y_test_named, y_pred_adjusted_named, labels=label_names, zero_division=0))
```

6.3 实验结果的分析比较

分类报告：

	precision	recall	f1-score	support
女	0.48	0.14	0.21	116
男	0.65	0.92	0.76	202
accuracy			0.63	318
macro avg	0.57	0.53	0.49	318
weighted avg	0.59	0.63	0.56	318

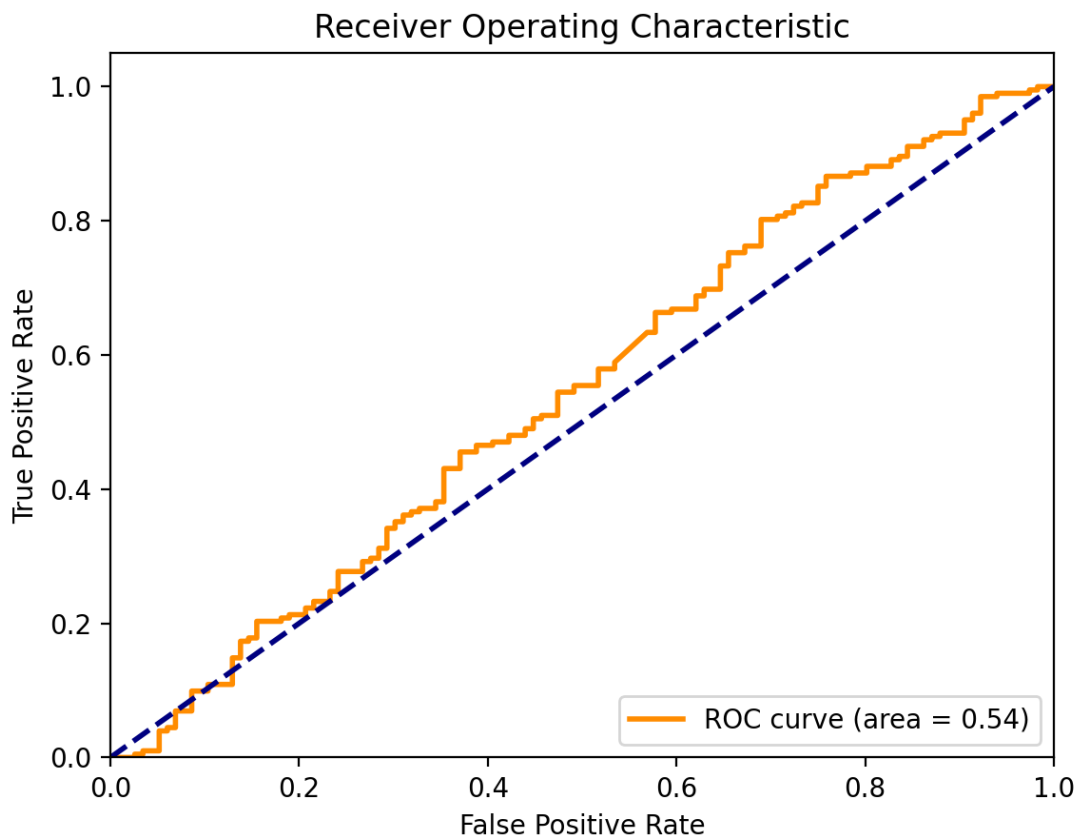
类别表现

女性：精确率 (Precision): 0.48 召回率 (Recall): 0.14 F1-score: 0.21 支持度 (Support): 116

男性：精确率 (Precision): 0.65 召回率 (Recall): 0.92 F1-score: 0.76 支持度 (Support): 202

总体性能指标准确率 (Accuracy): 0.63 宏平均 (Macro Average): 精确率: 0.57 召回率: 0.53 F1-score: 0.49

加权平均 (Weighted Average): 精确率: 0.59 召回率: 0.63 F1-score: 0.56



准确率 (Accuracy): 整体准确率为 0.63，这表明大约 63% 的测试样本被正确分类。然而，由于类别不平衡的存在，这个指标可能不能完全反映模型的真实性能。

宏平均 (Macro Average): 该指标考虑了每个类别的性能，而不考虑类别的频率。精确率、召回率和 F1-score 分别为 0.57、0.53 和 0.49，显示出模型在两个类别之间的平衡性较差，特别是在处理女性样本时。

加权平均 (Weighted Average): 加权平均考虑了每个类别的支持度，因此更能反映出实际应用中的性能。尽管如此，精确率、召回率和 F1-score 分别为 0.59、0.63 和 0.56，仍然表明存在改进的空间。

6.4 小结

通过上述参数设定和优化过程，我构建了一个基于支持向量机的人脸性别识别模型。尽管整体准确率为 63%，但在类别“男”的召回率达到了 92%，显示出模型在识别男性样本方面表现出色。然而，类别“女”的召回率较低（14%），这提示我们需要进一步优化特征工程或探索其他方法来提高女性样本的识别能力。为

了进一步提升模型的整体性能，特别是在处理类别不平衡和提高女性样本识别准确性方面，需要采取针对性的改进措施。未来的工作将集中在优化数据集、改进特征工程以及探索不同的模型架构上。

7 结论

在此次人脸识别-性别识别的项目中，我们分别采用了图卷积网络（GCN）、决策树、K-近邻算法（KNN）、支持向量机（SVM）和 BP 神经网络五种不同的算法，并结合图片数据进行了实验与比较。以下是对这些算法在图片数据处理方面的工作概述、优缺点比较以及项目总体需要改进的方面。

7.1 算法工作与优缺点比较

（1）图卷积网络（GCN）：

图片数据处理：GCN 在处理图片数据时，需要将图片数据转换为图结构。这通常涉及到将图片的像素或特征点视为图的节点，并根据它们之间的空间关系构建边。在人脸性别识别中，可以将人脸关键点视为节点，根据关键点的相对位置构建边。我们构建了 GCN 模型，用于处理图结构的人脸关键点数据，并进行性别分类。在数据处理阶段，将原始二进制数据转换为 JPG 图像，提取人脸关键点并转换为图数据。模型训练过程中，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。

性能表现：GCN 在捕捉节点间的复杂关系方面表现出色，适用于具有明确拓扑结构的数据。然而，在人脸性别识别任务中，由于图片数据的空间关系不如图数据那样明确，GCN 的性能可能受到一定影响。

优点：能够有效利用图结构信息，捕捉节点间的复杂关系。

缺点：对图片数据的转换过程复杂，且可能损失部分空间信息。

（2）决策树：

图片数据处理：决策树通常直接对图片的像素值或提取的特征进行处

理,无需特殊的图结构转换。在人脸性别识别中,可以从图片中提取如肤色、面部轮廓等特征作为输入。我们使用了决策树算法进行人脸性别分类。通过计算信息增益、基尼不纯度等指标选择最优特征进行分裂,生成决策树模型。在训练过程中,通过剪枝技术防止过拟合。

性能表现: 决策树算法简单直观,能够处理非线性关系,并提供特征的重要性分析。但在处理高维、复杂的图片数据时,可能需要较深的树结构,容易导致过拟合。

优点: 模型可解释性强,易于理解和实现。

缺点: 对噪声和数据分布不平衡敏感,且在大型数据集上可能表现不佳。

(3) K-近邻算法 (KNN):

图片数据处理: KNN 算法直接对图片的像素值或特征向量进行处理,无需训练过程。在人脸性别识别中,可以将每张图片的特征向量作为样本点,通过计算待分类图片与已知类别图片之间的距离进行分类。我们采用了 KNN 算法进行人脸性别分类。通过计算待分类样本与训练集中样本之间的距离,选择 K 个最近邻样本进行投票分类。在训练过程中通过交叉验证选择最优的 K 值。

性能表现: KNN 算法在人脸性别识别任务中表现出色,尤其是在选择合适的 K 值时,能够取得较高的准确率。然而,其计算成本较高,且对特征的尺度变化敏感。

优点: 算法简单,易于实现,且无需训练过程。

缺点: 计算量大,对特征尺度敏感,需要进行数据标准化处理。

(4) BP 神经网络:

图片数据处理: BP 神经网络可以直接对图片的像素值或提取的特征进行处理。在人脸性别识别中,可以将图片的像素值或 CNN 提取的特征作为输入,通过多层神经网络进行分类。我们构建了 BP 神经网络模型进行人脸性别分类。通过前向传播和反向传播不断调整网络的权值和阈值,优化模型

性能。训练过程中采用了数据标准化、隐藏层参数调优等技术。

性能表现：BP 神经网络在人脸性别识别任务中表现出良好的性能，通过优化网络结构和训练策略取得较高的准确率。然而，BP 神经网络对初始参数敏感，且可能陷入局部最优解。

优点：具有较强的学习能力和泛化能力，适用于复杂任务的分类和回归问题。

缺点：对初始参数敏感，可能陷入局部最优解，且训练时间较长。

(5) 支持向量机 (SVM)

图片数据处理：SVM 通常与特征提取方法结合使用，如使用卷积神经网络 (CNN) 提取图片的高级特征，然后将这些特征输入 SVM 进行分类。在人脸性别识别中，可以使用预训练的 CNN 模型提取人脸特征。我们构建了基于 SVM 的人脸性别识别模型，使用 RBF 核函数处理非线性问题，并通过网格搜索进行超参数调优。训练过程中采用了 SMOTE 技术处理类别不平衡问题。

性能表现：SVM 在处理二分类问题时表现出色，但在处理非线性关系和类别不平衡问题时可能需要额外的技巧，如使用核函数和 SMOTE 技术。在人脸性别识别任务中，SVM 对男性样本的识别准确率较高，但对女性样本的识别准确率较低。

优点：适用于小样本分类问题，且具有较高的分类准确率。

缺点：对大规模训练样本难以实施，对参数调节和核函数的选择敏感。

7.2 需要改进的方面

- 1) 数据预处理与增强：进一步优化数据预处理流程，提高数据质量。例如，可以采用更先进的数据增强技术来增加训练样本的多样性，提高模型的泛化能力。
- 2) 特征选择与提取：探索更有效的特征提取方法，如使用更先进的 CNN 模型或结合多种特征提取技术来提高特征的表示能力。

- 3) 模型选择与优化：针对不同类型的图片数据和任务需求选择合适的模型。例如，对于具有明确拓扑结构的数据可以选择 GCN 模型；对于二分类问题可以选择 SVM 或 BP 神经网络等。对模型进行优化，如调整网络结构、学习率、批量大小等超参数，或使用正则化、Dropout 等技术防止过拟合。
- 4) 集成学习与融合：探索集成学习方法，如结合多个模型的预测结果来提高整体准确率。例如，可以使用投票机制或加权平均等方法对多个模型的预测结果进行融合。
- 5) 性能评估与反馈：建立完善的性能评估体系，使用多种评估指标（如准确率、精确度、召回率、F1 分数等）对模型进行全面评估。根据评估结果及时调整模型参数和训练方法，形成闭环反馈机制以不断优化模型性能。

参考文献

1. Cunha, J.C., O.F. Rana, and P.D. Medeiros, *Future Trends in Distributed Applications and Problem-solving Environments*. 2005.
2. Thomas N. Kipf and Max Welling, Semi-Supervised Classification with Graph Convolutional Networks, arXiv, 1609.02907. 2017.<https://arxiv.org/abs/1609.02907.pdf>
3. 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>
4. 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.
5. Zhang Z, Cui P, Zhu W. Deep learning on graphs: A survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2020.
6. 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.
7. 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>
8. 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>
9. 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>

附录

项目源码

https://github.com/447662/Pattern_recognition_by_gdut_22_ic_class2/tree/main/PR_course_project_FR_sex

成员分工与贡献率

表 1 成员分工及贡献度自评

姓名	个人分工	小组任务贡献度
杨纯一	数据预处理、GCN算法	20%
王莺璇	RP神经网络算法、项目概览与结论	20%
程韵芝	支持向量机算法	20%
陈进宇	KNN算法	20%
陈柏壮	决策树算法	20%

课程体会与建议

在模式识别这门课程中，邢老师以严谨细致的教学态度和深入浅出的教学方法，让我们掌握了模式识别的基本原理和核心算法、领略到了这一领域的广阔前景和无限可能。课程的内容设置得合理且全面。从绪论部分对模式识别基本概念的介绍，到基于距离的分类器、朴素 Bayes 分类器、支持向量机等经典算法的学习，再到神经网络、决策树等高级主题，每一部分内容都环环相扣，层层递进，让我们逐步理解了这些算法的原理和应用，打下理论基础。

其次，课程非常注重理论与实践的结合。我们不仅通过课堂和华为云课学习了理论知识，还通过命题的翻转课堂和模式识别项目加深了对算法的理解和探索算法特点的能力，将学到的知识应用到实际问题中。通过小组内多种算法的分工运用和横纵比较事半功倍地令我们切实了解了五种算法的原理、特点及运用上的优劣。这种教学方式极大提高了我们的实操应用能力，还让我们在实践中发现了许多书本上没有的问题和挑战，进一步激发了我们的求知欲和探索欲，培养了我们的独立思考和解决问题的能力。

在学习过程中，我们也深刻感受到了模式识别这门课程的复杂性和挑战性。每一个算法都有其独特的优点和适用范围，但同时也存在一定的局限性和不足。因此，我们需要不断学习和探索新的算法和技术，以适应不断变化的需求和挑战。对于这门课程我们的建议是能够增加一些可靠的适合自主学习的课程资源和适

当增加编程实践环节的教学，这样接触项目中的算法时能够更快适应衔接。

总的来说，这一学期的模式识别课程让我们受益匪浅。十分感谢老师认真负责的教学指导和用心良苦的多样实践结合的课程设计。相信在未来的学习和工作中，这些经历将为我们提供有力的支持。