程序报告

学号：2213924　　姓名：申宗尚

一、问题重述

本实验采用特征脸（Eigenface）算法进行人脸识别。

特征脸（eigenface）是第一种有效的人脸识别方法，通过在一大组描述不同人脸的图像上进行主成分分析（PCA）获得。

本次实验构建一个自己的人脸库（建议）：可以选择基于 ORL 人脸库添加自己搜集到的人脸图像形成一个更大的人脸库，要求人脸库中的每一张图像都只包含一张人脸且眼睛的中心位置对齐(通过裁剪或缩放，使得每张人脸图像大小尺寸一致且人脸眼睛的中心位置对齐)。为了方便操作，也可以选择直接基于 ORL 人脸库进行本次实验。

在模型训练过程中，首先要根据测试数据求出平均脸，然后将前 K 个特征脸保存下来，利用这 K 个特征脸对测试人脸进行识别，此外对于任意给定的一张人脸图像，可以使用这 K 个特征脸对原图进行重建。

实验要求：

1. 求解人脸图像的特征值与特征向量构建特征脸模型
2. 利用特征脸模型进行人脸识别和重建，比较使用不同数量特征脸的识别与重建效果
3. 使用 Python 语言

二、设计思想

本次实验中，我们需要对三个主要函数进行编写（eigen_train，rep_face， rec_face）。特征脸算法采用 PCA 主成分分析方法，根据训练集生成特征脸，并且根据此进行测试集人脸的识别以及重建。

其中，最主要的设计如下：

1. 特征值降序排列，同时进行范围的限定，否则特征值运算时由于有矩阵运算，可能会导致维度不一致而报错。

2. 特征脸的数量选取。由于本次实验中，特征脸数量会影响最终人脸识别的准确率，对于选取的人脸数需要进行调参比较，在多次训练后，认为选取特征脸数量为 40 时，能达到最高的准确率 90%。

3. 在 PCA 主成分分析中，含有大量的运算、标准化、线性空间的转换等。

三、代码介绍

Eigen_train 函数：

```python
def eigen_train(trainset, k=40):
    """
    训练特征脸 (eigenface) 算法的实现
    :param trainset: 使用 get_images 函数得到的处理好的人脸数据训练集
    :param K: 希望提取的主特征数
    :return: 训练数据的平均脸, 特征脸向量, 中心化训练数据
    """
    #计算平均脸
    avg_img = np.mean(trainset, axis=0)
    #中心化数据
    norm_img = trainset - avg_img
    feature=trainset[0:k]
    #计算协方差矩阵
    cov_matrix = np.dot(norm_img.T, norm_img)
    #计算特征值和特征向量
    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
    # 对特征值进行降序排序, 并获取对应的索引
    idx=np.argsort(-eigenvalues)
    eigenvalues=eigenvalues[idx]
    eigenvectors=eigenvectors[:,idx]
    # 选择前 k 个最大的特征值对应的特征向量
    feature=eigenvectors[:,0:min(k,eigenvectors.shape[1])].T
    return avg_img, feature, norm_img
```

参数：

trainset：使用 get_images 函数得到的处理好的人脸数据训练集。

k：希望提取的主特征数，默认为 40。

返回值：

avg_img：训练数据的平均脸。

feature：特征脸向量。

norm_img：中心化训练数据。

首先，计算训练集的平均脸 avg_img，然后对训练集进行中心化处理，得到中心化的训练数据 norm_img。

接着，计算了训练数据的协方差矩阵，并通过特征值分解得到特征值和特征向量。

特征向量按照对应的特征值大小进行降序排序，并选择前 k 个最大的特征值对应的特征向量作为特征脸向量 feature。

最后返回了平均脸、特征脸向量和中心化训练数据。

Rep_face 函数：

```
def rep_face(image, avg_img, eigenface_vects, numComponents = 0):
    """
    用特征脸（eigenface）算法对输入数据进行投影映射，得到使用特征脸向量表示的数据

    :param image: 输入数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :return: 输入数据的特征向量表示，最终使用的特征脸数量
    """
    numEigenFaces=min(numComponents,eigenface_vects.shape[0])
    n=eigenface_vects[0:numEigenFaces,].T
    representation=np.dot(image-avg_img,n)
    return representation, numComponents
```

参数：

image：输入数据。

avg_img：训练集的平均人脸数据。

eigenface_vects：特征脸向量。

numComponents：选用的特征脸数量，默认为 0。

返回值：

representation：输入数据的特征向量表示。

numComponents：最终使用的特征脸数量。

该函数用于将输入数据投影到特征脸空间上，得到特征向量表示。

首先，根据输入的特征脸数量 numComponents，确定要使用的特征脸数量。

然后计算投影矩阵，将输入数据映射到特征脸空间上，得到特征向量表示 representation。

最后返回特征向量表示和实际使用的特征脸数量。

Rec_face 函数：

```
def recFace(representations, avg_img, eigenVectors, numComponents, sz=(112,92)):
    """
    利用特征人脸重建原始人脸

    :param representations: 表征数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :param sz: 原始图片大小
    :return: 重建人脸，str 使用的特征人脸数量
    """
    faces=np.dot(representations,eigenVectors[0:numComponents,])+avg_img
    return faces, 'numEigenFaces_{}'.format(numComponents)
```

参数：

representations：表征数据。

avg_img：训练集的平均人脸数据。

eigenVectors：特征脸向量。

numComponents：选用的特征脸数量。

sz：原始图片大小，默认为 (112, 92)。

返回值：

faces：重建的人脸。

numEigenFaces_{}"：使用的特征人脸数量的字符串形式。

这个函数用于利用特征脸重建原始人脸。

首先根据输入的特征脸数量，确定要使用的特征脸数量。

然后利用特征向量 representations 和特征脸向量 eigenVectors，加上平均脸 avg_img，通过线性组合的方式重建原始人脸。

最后返回重建的人脸和使用的特征人脸数量的字符串形式。

四、代码内容

```
# 在生成 main 文件时，请勾选该模块

# 导入必要的包
from sklearn.preprocessing import normalize
import matplotlib.pyplot as plt
import numpy as np
import cv2
from PIL import Image
import os

def spilt_data(nPerson, nPicture, data, label):
    """
    分割数据集

    :param nPerson : 志愿者数量
    :param nPicture: 各志愿者选入训练集的照片数量
    :param data : 等待分割的数据集
    :param label: 对应数据集的标签
    :return: 训练集，训练集标签，测试集，测试集标签
    """
    # 数据集大小和意义
    allPerson, allPicture, rows, cols = data.shape

    # 划分训练集和测试集
    train = data[:nPerson,:nPicture,:,:].reshape(nPerson*nPicture, rows*cols)
    train_label = label[:nPerson, :nPicture].reshape(nPerson * nPicture)
    test = data[:nPerson, nPicture:, :, :].reshape(nPerson*(allPicture - nPicture), rows*cols)
    test_label = label[:nPerson, nPicture:].reshape(nPerson * (allPicture - nPicture))
```

```
    # 返回: 训练集, 训练集标签, 测试集, 测试集标签
    return train, train_label, test, test_label

def plot_gallery(images, titles, n_row=3, n_col=5, h=112, w=92):   # 3 行 4 列
    """
    展示多张图片

    :param images: numpy array 格式的图片
    :param titles: 图片标题
    :param h: 图像 reshape 的高
    :param w: 图像 reshape 的宽
    :param n_row: 展示行数
    :param n_col: 展示列数
    :return:
    """
    # 展示图片
    plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
    plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)
    for i in range(n_row * n_col):
        plt.subplot(n_row, n_col, i + 1)
        plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)
        plt.title(titles[i], size=12)
        plt.xticks(())
        plt.yticks(())
    plt.show()

# 在生成 main 文件时, 请勾选该模块

def eigen_train(trainset, k=40):
    """
    训练特征脸（eigenface）算法的实现
    :param trainset: 使用 get_images 函数得到的处理好的人脸数据训练集
    :param K: 希望提取的主特征数
    :return: 训练数据的平均脸, 特征脸向量, 中心化训练数据
    """
    #计算平均脸
    avg_img = np.mean(trainset, axis=0)
    #中心化数据
    norm_img = trainset - avg_img
    feature=trainset[0:k]
    #计算协方差矩阵
    cov_matrix = np.dot(norm_img.T, norm_img)
    #计算特征值和特征向量
```

```
    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
    # 对特征值进行降序排序，并获取对应的索引
    idx=np.argsort(-eigenvalues)
    eigenvalues=eigenvalues[idx]
    eigenvectors=eigenvectors[:,idx]
    # 选择前 k 个最大的特征值对应的特征向量
    feature=eigenvectors[:,0:min(k,eigenvectors.shape[1])].T
    return avg_img, feature, norm_img

# 在生成 main 文件时，请勾选该模块

def rep_face(image, avg_img, eigenface_vects, numComponents = 0):
    """
    用特征脸（eigenface）算法对输入数据进行投影映射，得到使用特征脸向量表示的数据

    :param image: 输入数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :return: 输入数据的特征向量表示，最终使用的特征脸数量
    """
    numEigenFaces=min(numComponents,eigenface_vects.shape[0])
    n=eigenface_vects[0:numEigenFaces,].T
    representation=np.dot(image-avg_img,n)
    return representation, numComponents
def recFace(representations, avg_img, eigenVectors, numComponents, sz=(112,92)):
    """
    利用特征人脸重建原始人脸

    :param representations: 表征数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :param sz: 原始图片大小
    :return: 重建人脸, str 使用的特征人脸数量
    """
    faces=np.dot(representations,eigenVectors[0:numComponents,])+avg_img
    return faces, 'numEigenFaces_{}'.format(numComponents)
```

五、实验结果

自行训练准确率 90%，通过测试，完成任务要求。

```
训练数据集：（200，10304）
测试数据集：（80，10304）
特征脸数量：40
人脸识别准确率：90.0%
```

六、总结

本次实验采用特征脸（Eigenface）算法进行人脸识别，通过对训练集进行主成分分析（PCA），生成特征脸模型，并利用该模型对测试人脸进行识别和重建。实验要求包括求解人脸图像的特征值与特征向量构建特征脸模型，利用特征脸模型进行人脸识别和重建，并比较不同数量特征脸的识别与重建效果。

设计思想包括编写三个主要函数：eigen_train、rep_face、rec_face。在特征脸算法中，采用 PCA 方法生成特征脸，并在训练过程中对特征值进行降序排列和范围限定，同时选择合适的特征脸数量以提高识别准确率。在 PCA 主成分分析中，涉及大量运算、标准化和线性空间转换。

代码介绍包括三个主要函数的功能和参数介绍，以及代码实现。其中，eigen_train 函数用于训练特征脸模型，rep_face 函数用于对输入数据进行特征脸投影映射，rec_face 函数用于利用特征脸重建原始人脸。代码实现了数据集分割、图片展示、特征脸训练、人脸重建等功能。

实验结果显示自行训练的准确率达到了 90%，完成了任务要求。报告总结了实验过程中的收获，包括对模型的选择、参数调整、文本向量化方法的学习以及 Pipeline 的运用，提高了对机器学习、分类器、Python 编程等方面的熟练度。

据此，认为本次实验基本达到了预期目标，通过实践提升了对人脸识别算法的理解和应用能力，以及对 Python 编程和机器学习工具的熟练度。