**程序报告**

学号：2213410 姓名：徐俊智

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

====================================================================

本次实验的目标是实现特征脸（Eigenface）算法来进行人脸识别和人脸重建。特征脸算法是一种基于主成分分析（PCA）的方法，它通过分析和统计大量的人脸图像数据，提取出最能代表人脸变化的特征向量，即特征脸。实验的主要步骤包括数据准备、特征脸模型的构建、人脸识别模型的实现以及人脸重建模型的构建。

1. **设计思想**

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

====================================================================

1.数据预处理：对数据集进行预处理，包括分割训练集和测试集，以及将图像数据归一化。

2.训练特征脸：通过计算训练集图像的协方差矩阵，并对其执行特征值分解，来提取特征脸。选择前K个特征值对应的特征向量作为特征脸。

3.人脸识别：利用提取的特征脸对测试集图像进行投影，通过计算测试图像与训练集图像在特征空间中的距离，来识别测试图像的类别。

4.人脸重建：使用选定数量的特征脸来重建人脸图像，观察不同数量特征脸对重建效果的影响。

1. **代码内容**

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

====================================================================

# 导入必要的包

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()

datapath = './ORL.npz'

ORL = np.load(datapath)

data = ORL['data']

label = ORL['label']

num\_eigenface = 200

train\_vectors, train\_labels, test\_vectors, test\_labels = spilt\_data(40, 5, data,label)

train\_vectors = train\_vectors / 255

test\_vectors = test\_vectors / 255

def eigen\_train(trainset, k=20):

"""

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

"""

# 计算平均人脸

avg\_img = np.mean(trainset, axis=0)

# 对训练数据进行中心化处理

norm\_img = trainset - avg\_img

# 计算协方差矩阵

cov\_matrix = np.cov(norm\_img, rowvar=False)

# 计算特征值和特征向量

eigenvalues, eigenvectors = np.linalg.eigh(cov\_matrix)

# 根据特征值对特征向量进行排序

sorted\_indices = np.argsort(eigenvalues)[::-1]

eigenvectors = eigenvectors[:, sorted\_indices]

# 选择前 k 个特征向量

feature = eigenvectors[:, :k]

# 返回：平均人脸、特征人脸、中心化人脸

return avg\_img, feature, norm\_img

# 返回平均人脸、特征人脸、中心化人脸

avg\_img, eigenface\_vects, trainset\_vects = eigen\_train(train\_vectors, num\_eigenface)

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

"""

# 对输入图像进行中心化处理

centered\_img = image - avg\_img

# 投影到特征空间

if numComponents == 0:

numComponents = eigenface\_vects.shape[1]

representation = np.dot(centered\_img, eigenface\_vects[:, :numComponents])

numEigenFaces = numComponents

# 返回：输入数据的特征向量表示, 特征脸使用数量

return representation, numEigenFaces

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

"""

face = np.dot(representations, eigenVectors[:, :numComponents].T) + avg\_img

# 返回: 重建人脸, str 使用的特征人脸数量

return face, 'numEigenFaces\_{}'.format(numComponents)

1. **实验结果**

（实验结果，必填）

====================================================================



1. **总结**

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

====================================================================

在本次实验中，我成功实现了特征脸算法，并在测试集上达到了较高的识别准确率。

可能改进的方向：未来的工作可以探索更多的数据增强技术以提高模型的泛化能力，或者尝试结合深度学习方法来进一步提升识别和重建的性能。

实现过程中的困难：如何选择合适的特征脸数量以达到最佳的识别和重建效果。

提升性能：基于ORL人脸库继续添加人脸图像形成一个更大的人脸库，提高模型的泛化能力。