**程序报告**

学号：2311061 姓名：马淏怡

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）

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

1.1 实验背景

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。 本次实验要求大家构建一个自己的人脸库（建议）：大家可以选择基于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)。为了方便同学们操作，大家也可以选择直接基于ORL人脸库进行本次实验。

1.2 实验内容

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。

1.3 实验要求

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，比较使用不同数量特征脸的识别与重建效果

使用 Python 语言

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**）

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

首先将每张二维人脸图像展开为一个一维向量，随后对这些向量应用特征提取算法。

输入：包含 n 个 1024 维人脸样本的一组矩阵，以及目标降维后的维度。

输出：一个映射矩阵，其中每一列表示一个特征人脸。

处理流程如下：

对每一个人脸样本向量进行中心化处理，即减去所有样本的平均向量；

计算中心化后的样本集合的协方差矩阵；

对协方差矩阵进行特征分解，并按照特征值从大到小排序；

选取前 l 个最大特征值对应的特征向量，组合成最终的映射矩阵；

将每个人脸向量投影到该映射矩阵上，实现降维。

每幅人脸图像都可通过与特征人脸矩阵相乘，得到一个包含 l 个分量的系数向量，表示该图像在特征空间中的坐标。换句话说，任何一张人脸图像都可以用这些特征人脸的线性组合进行表示。对任意一张待识别人脸，也可用相同方法获得其低维表示，再与数据库中所有人脸的低维向量进行相似度比对，从而识别出最接近的匹配。

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）

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

1. 训练特征脸

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

"""

###############################################################################

#### 训练特征脸（eigenface）算法的实现 ####

#### 请勿修改该函数的输入输出 ####

###############################################################################

# #

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

avg\_img = avg\_img/trainset.shape[0] #获得平均人脸(10304,)

differential\_matrix = trainset - avg\_img #获得差值矩阵(200,10304)

norm\_img = differential\_matrix #中心化人脸即差值矩阵

differential\_matrix\_t = differential\_matrix.T #差值矩阵的转置(10304, 200)

covariance\_matrix = np.mat(differential\_matrix) \* np.mat(differential\_matrix\_t) #获得协方差矩阵的转置(200,200)

eigenvalue, featurevector = np.linalg.eig(covariance\_matrix) #获得转换之后的特征值、特征向量

k\_min = min(featurevector.shape[1], k) #确保k的取值不会超限

feature\_vector = np.mat(differential\_matrix\_t) \* np.mat(featurevector[:,:k\_min]) #原始的特征向量(10304,20)

feature = np.mat(differential\_matrix\_t) \* np.mat(differential\_matrix) \* np.mat(feature\_vector)

#获得前二十个特征人脸

feature = np.array(feature).T

# #

###############################################################################

############# 在生成 main 文件时, 请勾选该模块 #############

###############################################################################

# 返回：平均人脸、特征人脸、中心化人脸

return avg\_img, feature, norm\_img

1. 人脸识别

# 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

"""

###################################################################################

#### 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 ####

#### 请勿修改该函数的输入输出 ####

###################################################################################

# #

difference\_image = np.array(image) - np.array(avg\_img) #得到差值图像(10304,)

num = min(eigenface\_vects.shape[0], numComponents)

eigenface\_vect = normalize(np.array(eigenface\_vects[:num, :]), norm = 'l2') #此时得到的归一化的矩阵(20,10304)

linear\_space = eigenface\_vect.T #此时得到人脸的线性空间(10304, 20)

coordinate = np.mat(difference\_image) \* np.mat(linear\_space) #得到坐标(10304,)\*(10304, 20) = (20,)

representation = coordinate #得到特征向量表示的人脸(20,)

numEigenFaces = numComponents

# #

###################################################################################

############# 在生成 main 文件时, 请勾选该模块 #############

###################################################################################

# 返回：输入数据的特征向量表示, 特征脸使用数量

return representation, numEigenFaces

1. 人脸重建

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

"""

###############################################################################

#### 利用特征人脸重建原始人脸 ####

#### 请勿修改该函数的输入输出 ####

###############################################################################

# #

num = min(numComponents, eigenVectors.shape[0]) #获得符合规范的特征脸数量

eigenface\_vects = np.array(eigenVectors[:num, :]) #根据需要的数量获得对应的矩阵(20,10304)

eigenface\_vects = normalize(eigenface\_vects, norm = 'l2') #二范式归一化

linear\_space = eigenface\_vects.T #获得坐标转换的线性空间

matrix = np.mat(linear\_space) \* np.mat(linear\_space).T #先两边都右乘一个线性空间的转置，变为方阵

matrix\_inversion = np.linalg.inv(np.mat(matrix)) #求该方阵的逆矩阵

face = np.mat(representations) \* np.mat(linear\_space).T \* np.mat(matrix\_inversion) #获得face的差值图片

face = np.array(face) + np.array(avg\_img) #获得face的原始图片

face = letterbox\_image(face, sz)

# #

###############################################################################

############# 在生成 main 文件时, 请勾选该模块 #############

###############################################################################

# 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**）

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

1. 协方差矩阵计算可优化
2. 人脸重建方式可优化
3. 参数选择

k选择了20，但是20个特征覆盖率可能不足

人脸尺寸，目前是缩放，可能导致图像变形信息缺失

1. 可以抽象为类，如 EigenFaceRecognizer
2. 可以在相同框架下对比比如CNN、PCA等方法的性能