程序报告

学号：2211290　　姓名：姚知言

一、问题重述

结合论文和维基百科，采用特征脸（Eigenface）算法基于 ORL 人脸库进行人脸识别，求解人脸图像的特征值和特征向量构成特征脸模型，并使用该模型分别完成人脸识别和人脸重建两个子实验，观察实验效果与数据集选择大小的关系。

二、设计思想

首先完成特征脸算法的实现。该部分需要返回三个参数，分别为特征人脸，平均人脸，中心化人脸。平均人脸即为对训练数据按列取平均值，使用 np.mean（np 即为 numpy 库，下同）实现。将原训练集人脸各维度数据对应减去平均值，所得即为中心化人脸。由于每张图片中维度较大，对原矩阵求取特征值和特征向量开销较大，在图片数量较少的时候通过另一种方法可以仅仅在以图片数量为矩阵中进行计算。具体实现过程为：先将中心化人脸乘以自身转置（记为 corr），并对其求取特征值和特征向量（使用 np.linalg.eig）。将中心化人脸的转置乘以（按照数量要求筛选后的，筛选参照输入 k 和原本数据量的较小值）特征向量，所得结果即为题设的特征向量（记为 values）。然后将中心化人脸的转置乘以中心化人脸再乘以 values，就可以得到所需的特征脸向量。根据函数数据要求需要将该矩阵转置后输出，我们可以简单的应用线性代数的知识化简上式，所得即为该部分最后一行 feature 特征脸向量的计算方式。

然后完成人脸识别模型。首先确定特征脸使用数量（numEigenface），即为输入 numComponents 和特征脸向量的行数的较小值。然后用特征脸向量的前 numEigenface 行乘以[输入数据与平均脸之差]，再将所得结果转置，即为输入数据的特征向量表示。但这样实现的识别模型测试结果并不理想，正确率仅仅为 60%多。根据分析，发现是未将特征脸向量进行归一化导致的，所以在特征脸向量矩阵使用前先将其进行欧几里得归一化，优化后的模型正确率为 90%。

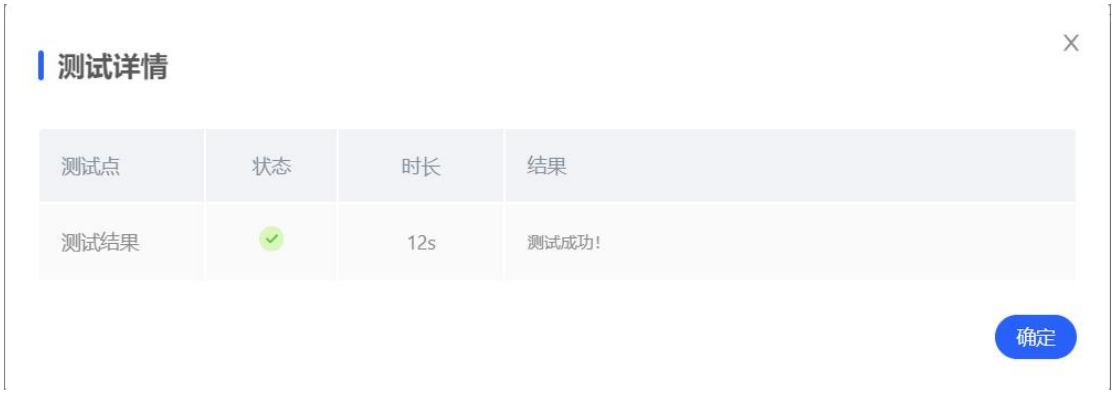
最终完成人脸重建模型。事实上人脸重建模型即为人脸识别模型的逆过程。首先和上一部分相同，选取特征脸向量的 numComponents（输入和特征脸向量行数的较小值）行，并进行归一化。通过上一部分的公式可以得到原始人脸（face）的计算表达式为处理后特征脸向量的伪逆（np.linalg.pinv）乘上表征数据的转置，再加上平均脸。最终将一维向量根据 sz 参数转换为二维向量，以便后续调用图片输出函数进行输出。根据运行结果可以很清晰的看出人脸重建模型较为成功，且随着选择的特征脸向量行数的增加图片的清晰度和拟合度也在逐渐增加。

三、代码内容

```
#特征人脸算法
def eigen_train(trainset, k=20):
    avg_img=np.mean(trainset,axis=0)
    norm_img=trainset-avg_img
    corr=norm_img@norm_img.T
```

```
    eigenvalues,eigenvectors = np.linalg.eig(corr)
    values=norm_img.T@eigenvectors[:,:min(k,trainset.shape[0])]
    feature=values.T@norm_img.T@norm_img
    return avg_img, feature, norm_img
==========================================================================
#人脸识别模型
from sklearn.preprocessing import Normalizer
def rep_face(image, avg_img, eigenface_vects, numComponents = 0):
    numEigenFaces=min(numComponents,eigenface_vects.shape[0])
    #归一化测试组
    normalizer = Normalizer(norm='l2')
    normalized_vectors = normalizer.transform(eigenface_vects[:numEigenFaces,:])
    representation=(normalized_vectors@(image-avg_img)).T
    #保留的未进行归一化测试组
    #representation=((eigenface_vects[:numEigenFaces,:])@(image-avg_img)).T
    return representation, numEigenFaces
==========================================================================
#人脸重建模型
def recFace(representations, avg_img, eigenVectors, numComponents, sz=(112,92)):
    numComponents=min(numComponents,eigenVectors.shape[0])
    normalizer = Normalizer(norm='l2')
    normalized_vectors = normalizer.transform(eigenVectors[:numComponents,:])
    face=np.linalg.pinv(normalized_vectors) @ representations.T+avg_img
    face.reshape(sz)
    return face, 'numEigenFaces_{}'.format(numComponents)
```

四、实验结果



五、总结

在本次实验中，通过对特征脸算法的学习，我复习了主成分分析（PCA）算法的主要实现过程，也增进了阅读英文科学论文的经验。对于特征脸算法的实现过程以及使用方式也有了较为全面的理解。同时，在对特征脸算法的复现过程中，通过 numpy 库向量操作对矩阵求转置，求逆，求取特征根特征向量，以及求取伪逆等操作均有了较好的认识，并对 sklearn 库中欧几里得归一化的方式进行了熟悉。通过数学公式的推导和代码编写成功解决了问题，也

使得我倍感收获。在本次实验中，遇到了一些因为不熟悉矩阵操作和题目要求的输入输出导致矩阵规模不匹配的情况，但都根据.shape 的方式增加对数据的了解完成化解。本次实验的不足之处包括没有使用更大的人脸训练数据集，若使用更大的数据集或许可以获得更好的算法实现。