程序报告
一、问题重述
二、设计思想
三、代码内容
数据集的加载
训练特征脸算法的实现
人脸识别模型
人脸重构
四、实验结果

程序报告

学号：2312236　　姓名：付家权

一、问题重述

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

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

首先我们需要把所有二维的人脸$squeeze$为一个一维的向量，$row \times col \to 1 \times (row \times col)$，然后进行如下算法：

- 输入： n个1024维人脸样本数据所构成的矩阵X，降维后的维数l
- 输出：映射矩阵$W = w_1, w_2, \ldots, w_l$ (其中每个$w_j (1 \leq j \leq l)$是一个特征人脸

1. 对于每个人脸样本数据$x \times i$进行中心化处理：进行中心化处理：$x_i = x_i - \mu, \mu = \frac{1}{n} \sum \times j = 1^n x_j$
2. 计算原始人脸样本数据的协方差矩阵:$\sum = \frac{1}{n-1} X^T X$
3. 对协方差矩阵进行特征值分解，对所得特征根从到小排序$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
4. 取前l个最大特征根所对应特征向量$w_1, w_2, \ldots, w_l$组成映射矩阵W
5. 将每个人脸图像$x \times i$按照如下方法降维按照如下方法降维:$(x_i) \times 1 \times d(W)_{d \times l} = 1 \times l$

将每幅人脸分别与每个特征人脸做矩阵乘法，得到一个相关系数向量，包含l个系数，这个向量表示我们可以用l个特征人脸的线性组合表示任意一个人脸数据。对于任意一个未知人脸，我们都可以得到其降维后的向量，然后将这个向量和数据库中的所有人脸向量进行对比，找到最接近的向量，从而进行分类。

三、代码内容

数据集的加载

```
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

datapath = './ORL.npz'
ORL = np.load(datapath)
data = ORL['data']
label = ORL['label']
```

```
print("数据格式(志愿者数, 各志愿者人脸数, height, width):", data.shape)
print("标签格式(志愿者数, 各志愿者人脸数):", label.shape)

train_vectors, train_label, test_vectors, test_label = spilt_data(40, 5, data,label)
print("训练数据集:", train_vectors.shape)
print("测试数据集:", test_vectors.shape)
```

该部分完成了我们本次实验的数据集的加载，我们打开了我们的数据路径`ORL.npz`，然后分别读取我们的数据和标签，将其存储到我们的`train_vectors`，`train_label`，`test_vectors`，`test_label`中，这样我们就可以开始后面的训练了。

训练特征脸算法的实现

```
def eigen_train(trainset, k=20):
    avg_img=np.mean(trainset,axis=0)
    norm_img=trainset-avg_img
    feature=trainset[0:k]
    cov_matrix=np.dot(norm_img.T,norm_img)
    eigenvalues,eigenvectors=np.linalg.eigh(cov_matrix)
    idx=np.argsort(-eigenvalues)
    eigenvalues=eigenvalues[idx]
    eigenvectors=eigenvectors[:,idx]
    feature=eigenvectors[:,0:min(k,eigenvectors.shape[1])].T
    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)

# 打印两张特征人脸作为展示
eigenfaces = eigenface_vects.reshape((num_eigenface, 112, 92))

eigenface_titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]
plot_gallery(eigenfaces, eigenface_titles, n_row=1, n_col=2)
```

此处我们按照课堂上所讲述的方法来实现我们的训练特征脸算法

1. 对于每个人脸样本数据$x \times i$进行中心化处理：进行中心化处理：$x_i = x_i - \mu, \mu = \frac{1}{n} \sum \times j = 1^n x_j$

    ```
    avg_img=np.mean(trainset,axis=0)
    norm_img=trainset-avg_img
    feature=trainset[0:k]
    ```

    可以看到此处我们将我们的平均人脸先进行聚类，然后再用我们的测试人脸减去我们的平均人脸，最后取前k个特征即可

2. 计算原始人脸样本数据的协方差矩阵:$\sum = \frac{1}{n-1} X^T X$

```
    cov_matrix=np.dot(norm_img.T,norm_img)
```

此处我们完成了协方差矩阵的计算

3. 对协方差矩阵进行特征值分解，对所得特征根从到小排序$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$

```
    eigenvalues,eigenvectors=np.linalg.eigh(cov_matrix)
    idx=np.argsort(-eigenvalues)
```

此处我们完成了特征值分解，然后将其按照倒序进行排序，所以我们取-1的索引值

4. 取前l个最大特征根所对应特征向量$w_1, w_2, \ldots, w_l$组成映射矩阵W

```
    eigenvalues=eigenvalues[idx]
    eigenvectors=eigenvectors[:,idx]
```

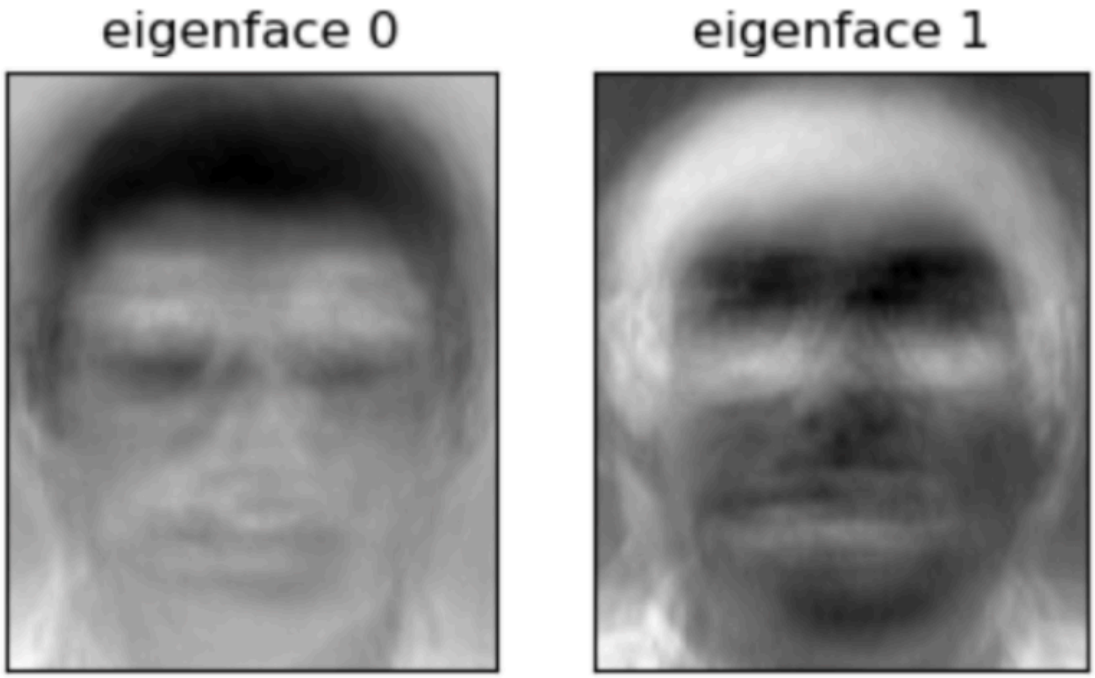
此处我们完成了取前l个最大的特征根，我们采用切片的方式去求解

5. 将每个人脸图像$x \times i$按照如下方法降维按照如下方法降维:$(x_i) \times 1 \times d(W)_{d \times l} = 1 \times l$

```
    feature=eigenvectors[:,0:min(k,eigenvectors.shape[1])].T
```

此处我们完成了我们最后的降维工作，采用了min直接进行降维

最后我们输出了我们的两张特征人脸



人脸识别模型

```
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])
    w=eigenface_vects[0:numEigenFaces,].T
    representation = np.dot(image-avg_img,w)

    # 返回：输入数据的特征向量表示，特征脸使用数量
    return representation, numEigenFaces

train_reps = []
for img in train_vectors:
    train_rep, _ = rep_face(img, avg_img, eigenface_vects,num_eigenface)
    train_reps.append(train_rep)

num = 0
for idx, image in enumerate(test_vectors):
    label = test_labels[idx]
    test_rep, _ = rep_face(image, avg_img, eigenface_vects, num_eigenface)

    results = []
    for train_rep in train_reps:
        similarity = np.sum(np.square(train_rep - test_rep))
        results.append(similarity)
    results = np.array(results)

    if label == np.argmin(results) // 5 + 1:
        num = num + 1

print("人脸识别准确率：{}%".format(num / 80 * 100))
```

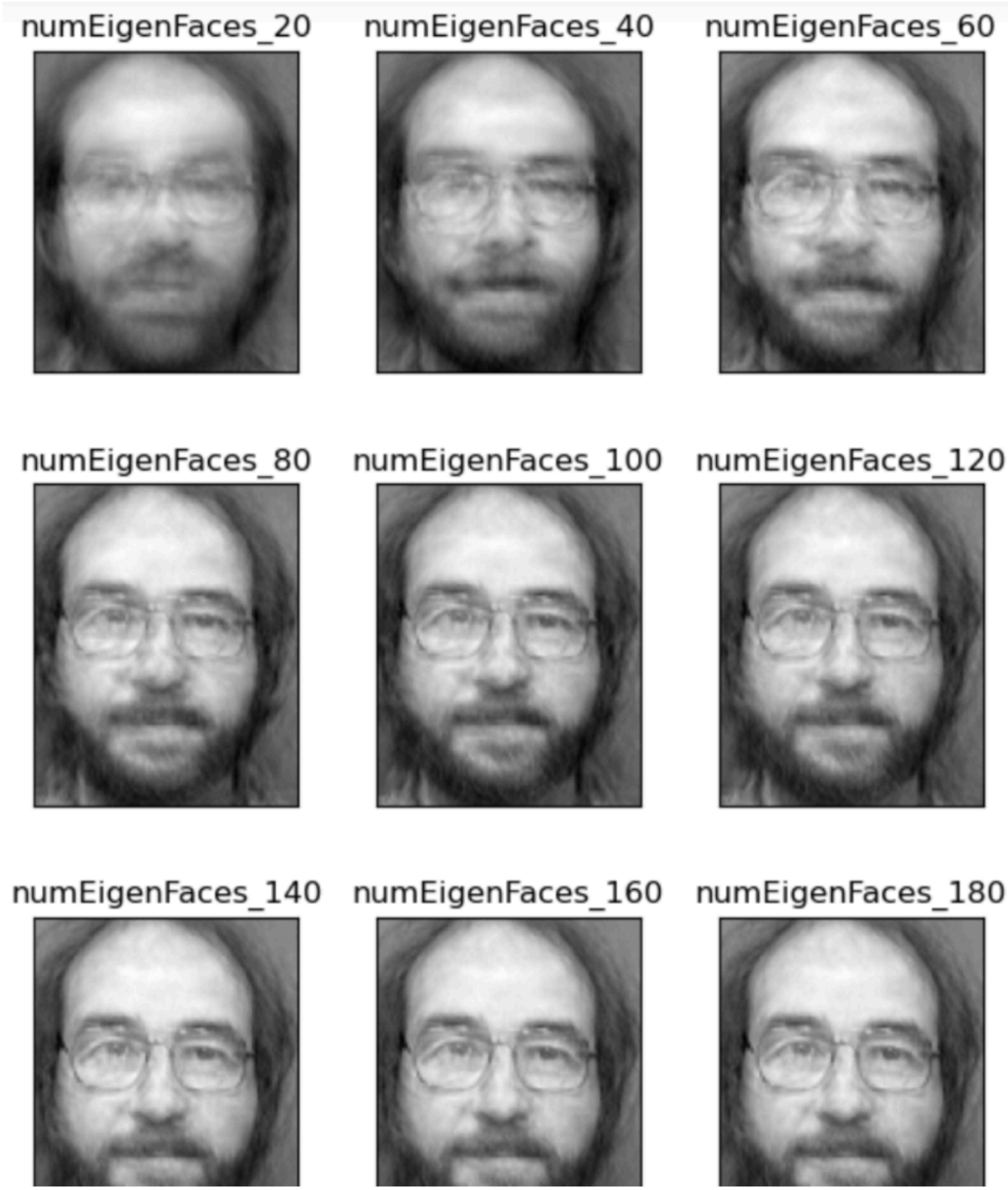
此处我们使用特征脸（$Eigenface$）算法对测试集中的人脸照片进行预测，我们在这里定义了 `rep_face` 函数，其输入是测试数据, 训练集的平均人脸数据，特征脸向量, 选用的特征脸数量，最后输出我们的人脸识别的准确率，最后我们也是达到了91.25%的正确率！

人脸重构

```
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[0:numComponents,])+avg_img

    # 返回: 重建人脸, str 使用的特征人脸数量
    return face, 'numEigenFaces_{}'.format(numComponents)

print("重建训练集人脸")
# 读取train数据
image = train_vectors[100].copy()

faces = []
names = []
# 选用不同数量的特征人脸重建人脸
abc=[]
#faces.append(image)
for i in range(20, 200, 20):
    representations, numEigenFaces = rep_face(image, avg_img, eigenface_vects, i)
    face, name = recFace(representations, avg_img, eigenface_vects, numEigenFaces)
    faces.append(face)
    names.append(name)

plot_gallery(faces, names, n_row=3, n_col=3)

print("-"*55)
print("重建测试集人脸")
# 读取test数据
image = test_vectors[54].copy()

faces = []
names = []
#faces.append(image)
# 选用不同数量的特征人脸重建人脸
for i in range(20, 200, 20):
    representations, numEigenFaces = rep_face(image, avg_img, eigenface_vects, i)
    face, name = recFace(representations, avg_img, eigenface_vects, numEigenFaces)
    faces.append(face)
    names.append(name)

plot_gallery(faces, names, n_row=3, n_col=3)
```

此处我们完成了在人脸识别任务的前提下，对我们的模型进行重构，我们将我们训练出来的特征人脸再加上我们每个人的特征值，这样就可以还原出我们原来的照片了！我们对其分别进行测试，发现我们的还原出来的图片是比较符合原来的图像的，说明我们的模型是比较优秀的。



四、实验结果

我们将我们的各项代码放在$main$文件中进行测试，结果显示正确！

测试详情

测试点	状态	时长	结果
测试结果	✓	320s	测试成功!

确定

综上所述，我们本次人脸识别的实验就成功完成了！