深 圳 大 学 实 验 报 告

课程名称：智能识别系统设计

实验项目名称：视频人脸特征提取

学院：计算机与软件学院

专业：软件工程（腾班）

指导教师：沈琳琳、文嘉俊

报告人：洪子敬 学号：2022155033 班级：腾班

实验时间：2024 年 9 月 29 日至 11 月 2 日

实验报告提交时间：2024 年 10 月 31 日

教务处制

一、实验目的与要求：

实验目的：

(1) 编程读取人脸视频或图像序列并实现对人脸的检测与预处理（提示：人脸检测方法可采用 AdaBoost、人脸关键点提取等机器学习方法）；

(2) 编写 PCA 或其他特征提取算法对人脸图像进行特征提取；

(3) 设计并构建人脸识别框架。

(4) 对参数进行实验分析。

实验要求：

(1) 熟悉 OpenCV 编程环境或其他图像处理工具包；

(2) 熟练掌握基本图像处理方法和 PCA 子空间学习等人脸特征提取方法；

(3) 熟悉最近邻等图像分类方法；

(4) 编程语言推荐使用 C++，但不限于 C++（例如可使用 Python 或 Matlab 语言）；

(5) 无需编写界面，代码写成函数，方便调用（若以 C++为编程语言，建议以类的形式把相关函数统一起来，方便后续使用）。

二、实验内容与方法：

(1) 从视频或图像序列中创建训练样本，其中类别个数为 10，每类样本有 10 个人脸图像；

(2) 采用 PCA 或其他人脸特征提取方法提取人脸特征，并存储到文本文件中（提示：若用 PCA 方法，则需要把训练样本投影到子空间，获得训练样本在子空间的投影以及投影矩阵），以便测试时调用；

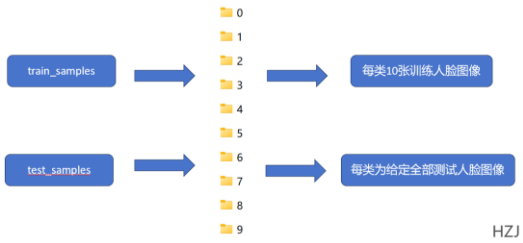
(3) 对 10 个类别的测试视频或测试图像序列进行测试，对于每一个类，输入其测试视频或测试图像序列，然后通过人脸检测、归一化、分类等一系列方法实现对视频或图像序列中人脸的识别，并统计识别率，若 10 个测试视频（或图像序列）都识别正确，则识别率为 100%。

(4) 对模型稳定性造成影响的参数进行实验分析。

三、实验步骤与过程：（其中：提供有简短说明的程序代码。要求：程序运行正确、符合设计要求。）

1. 数据预处理

根据实验要求可知，首先需要从给定的图像序列中创建训练样本，每类 10 张人脸图像，一共 10 类；测试图像只需要使用给定的全部 test 样本即可。为了实验时的方便，我们采用以下的图像存放格式：



通过这种存放格式，我们只需要通过目录的叠加形式获取每张图片，同时还能获得每张图片的真实标签，方便计算准确率。

此外，由于给定的训练样本有很多，而要求只能使用 10 图像进行训练，所以手动挑选了每个类具有代表性的图像 10 张，包括模糊的、遮挡的、光线明暗以及位置偏差的和人脸表情等具有代表性特征差异的图像组成各自的训练样本。

2. 人脸检测

本实验已经给出了用于人脸检测的分类器模型训练好的模型参数，我们可以直接加载，并用其来识别图像中的人脸。

而这个模型是 Haar 特征分类器（Haar Cascade Classifier），通过使用一系列的特征（称为 Haar 特征）来快速检测对象；它通过将图像分成多个区域并计算这些区域的特征，来判断这些区域是否包含目标对象（在这里是人脸）。

下面简单介绍 Haar 特征分类器（Haar Cascade Classifier）：

（1）Haar 特征：一种简单的矩形特征，用于表示图像的局部区域；基本的 Haar 特征可以分为：边缘特征（表示图像中亮度变化的边缘）、线性特征（表示图像中亮度变化较大的区域）和四方特征（通过对比两个区域的亮度值来检测特定的形状）。

（2）特征提取：使用图像积分，即使用积分图（Integral Image）来快速计算 Haar 特征。积分图是一种预处理图像的方式，在 O(1)的时间内能计算任意矩形区域的像素总和。

（3）分类器训练：通过机器学习的方法进行训练，具体包括以下步骤：

a. 正负样本：准备包含目标对象（如人脸）的正样本和不包含目标对象的负样本；

b. 特征选择：使用 AdaBoost 算法从所有可能的 Haar 特征中选择出最佳特征；

c. 构建分类器：将选择的特征组合成一个强分类器。AdaBoost 会创建一个由多个弱分类器组成的强分类器，这些弱分类器通过简单的特征来判断目标是否存在。

（4）级联分类器：Haar Cascade 是将多个分类器组合在一起形成的一个级联结构。每个级联级别都包含一个分类器，逐步过滤掉不太可能包含目标对象的区域。主要包括：

a. 多级检测（每个级别的分类器会对输入区域进行判断，如果分类器认为该区域不包含目标对象，则直接丢弃该区域，减少后续计算）；

b. 加速检测（通过逐级过滤，级联分类器可以快速排除大部分不包含目标的区域，从而提高检测速度）。

（5）目标检测：使用训练好的模型进行目标检测的主要步骤包括：

a. 图像传入：将待检测的图像传入级联分类器；

b. 滑动窗口：在图像上以滑动窗口的方式进行检测，窗口大小可以不同，以检测不同大

小的目标。

c. 多尺度检测：在不同的图像尺度上应用级联分类器，检测人脸或其他目标。

d. 返回结果：检测到的区域以矩形框的形式返回。

（6）优缺点：

a. 优点：算法通过级联结构，较为快速地排除无关区域得出结果并在各种干扰条件下（如光照条件变化、观察角度和模糊程度等）表现良好；

b. 缺点：如果目标被遮挡，可能会导致检测失败，即对**遮挡敏感**。

在 opencv 中已经定义了用于使用和加载 Haar 特征分类器的类 CascadeClassifier，通过加载训练好的模型参数，我们可以对给定图片进行人脸检测，并将检测出的矩形区域进行**裁剪、缩小尺寸和拉平后**放置在预定好的训练或者测试样本中，便于后续的 PCA 训练和测试。详细代码如下所示：（包括了数据预处理和人脸检测两部分）

```
# 加载人脸检测模型
# face_mode = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_alt(2).xml')
face_mode = cv2.CascadeClassifier("D:/tmp_data/haarcascade_frontalface_alt(2).xml")

# 训练集和测试集
train_path = "train_samples1"
train_images = []
train_labels = []

# 人脸检测
for label in os.listdir(train_path):
    label_path = os.path.join(train_path, label)
    for img_name in os.listdir(label_path):
        img_path = os.path.join(label_path, img_name)
        img = cv2.imread(img_path)

        if img is not None:
            gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_mode.detectMultiScale(gray_img, scaleFactor=1.1, minNeighbors=5)
            print(f"Image: {img_name}, Detected faces: {len(faces)}")
            for (x, y, w, h) in faces:
                face = gray_img[y:y+h, x:x+w]
                face_sized = cv2.resize(face, dsize: (50, 50))
                train_images.append(face_sized.flatten())
                train_labels.append(label)

# 将训练数据转为numpy数组
X_train = np.array(train_images)
y_train = np.array(train_labels)
```

（由于训练和测试的人脸图像检测过程类似，这里只展示训练时的人脸检测和数据处理）

此外，这里使用训练好的模型进行图像人脸检测时调用了 detectMultiScale 方法，输入的共 3 个参数：gray_img 为输入图像、scaleFactor 指定了图像尺寸的缩放比例（每次缩放一定比例，较小的值对检测到更多物体，但会增加计算量）和 minNeighbors 指定每个候选矩形需要保留的邻居数量，通俗来讲，检测到的每个矩形至少需要有多少个邻居矩形（即重叠部分大于某个阈值）才能被认为是有效的人脸。后两个参数是影响检测的效果的关键因素，是需要调好的。

3. PCA 进行特征降维

（1）原理简介：PCA（Principal Component Analysis）是一种常用的降维技术，旨在通过线性变换（找到一个新的坐标系，其中第一主成分是数据中方差最大的方向，第二主成分是与第一主成分正交的方向，依此类推）将高维数据投影到低维空间，同时尽可能保留数据的方差和结构信息。

（2）主要步骤包括：

a. 标准化数据：对数据进行中心化和标准化，确保每个特征的均值为 0，方差为 1；

b. 计算协方差矩阵：找出不同特征之间的关系；

c. 特征值分解：对协方差矩阵进行特征值分解，获取特征值和特征向量；

d. 选择主成分：按照特征值大小进行排序，选择前 k 个主成分；

b. 转换数据：将原始数据投影到选定的主成分上，得到降维后的数据。

（3）优缺点：

a. 对数据进行降维，减少计算负担，提高模型的可解释性和性能；

b. 仅适用于线性关系,可能丢失重要的非线性信息。

（4）代码实现

代码函数中数据输入是**按列排放**，首先对每个特征取均值，对每个特征进行中心化，并计算数据的协方差矩阵；再对其进行特征分解得到特征值和特征向量，并利用特征值大小降序排序，得到对方差贡献率的排序，并选择前 k 个特征值对应的特征向量组成新的投影矩阵（这里的 k 实际上就是下降后的特征维数）。

不过要注意的是在计算协方差矩阵时，可能会出现复数，会导致后面的人脸识别时出现数据异常，于是这里使用 numpy 库中给定的 real 函数，提取数据的实部，问题解决。

详细代码如下所示：

```
# 数据格式为按列
1 usage
def PCA(X_train, components):
    # Prepare
    col = X_train.shape[1]
    # axis=0 表示操作是针对行的方向，但实际上是在处理特征（列）
    Mean = np.mean(X_train, axis=1)
    # X = X_train - Mean
    # 减去每个特征的均值,使得每个特征的均值均为0
    X = X_train - Mean[:, np.newaxis]
    # Calculate eig_vectors, eig_values
    cov = np.dot(X, X.T) / col
    eigenvalues, eigenvectors = np.linalg.eig(cov)
```

```
# 确保特征值、特征向量为实数
eigenvalues = np.real(eigenvalues)
eigenvectors = np.real(eigenvectors)

# 对特征值、特征向量进行排序
index = np.argsort(eigenvalues)[::-1]
# 大小为样本数量*样本数量
W = eigenvectors[:, index]

# Build model
model = {
    'mean': Mean,
    'W': W[:, :components],
    'components': components
}
```

4.KNN 算法进行人脸识别分类

（1）原理简介：K-最近邻（K-Nearest Neighbors，KNN）是一种简单而有效的监督学习算法，常用于分类和回归任务；KNN 不需要假设数据分布，属于懒惰学习（lazy learning）方法，模型在训练阶段不进行显式学习，而是在预测时直接使用训练数据；此外，KNN 是根据样本之间的距离（通常是欧氏距离）来进行分类或回归。

（2）算法步骤包括：

a. 选择参数 K：选择一个正整数 K，表示在进行预测时考虑的邻居数量；

b. 计算距离：对于待分类的样本，计算其与所有训练样本的距离；

c. 找到最近邻：选择 K 个距离最近的训练样本；

d. 投票或平均：（本实验室分类问题，采用第一种）

分类问题：对 K 个邻居的类别进行投票，选择出现频率最高类别作为预测结果；

回归问题：对 K 个邻居的值取平均，作为预测结果。

（3）优缺点：

a. 不需要训练阶段，适合小规模数据集，可以处理多分类问题；

b. 对于大规模数据集，计算成本高，效率低，同时对噪声和 K 值选择较为敏感。

（4）代码实现

由于 KNN 的实现也较为简单，这里简单调用 sklearn 库函数进行实现。首先是定义 K 为 3，并用 PCA 后的训练数据和训练标签进行 KNN“训练”（实际上是数据存储）；接着把测试数据经过 PCA 投影矩阵进行特征降维，并将其传入 KNN 进行标签预测，最后利用 sklearn 中 accuracy_score 函数来计算最后的准确率并打印出来。

详细代码如下：

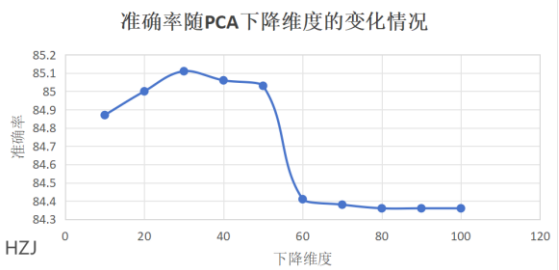
```
# KNN训练和识别
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train_pca, y_train)

# x_test_pca = pca.transform(X_test)
x_test_pca = X_test.dot(model['W'])

predictions = knn.predict(x_test_pca)

accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy is {accuracy*100:.2f}")

cv2.destroyAllWindows()
```

5. 实验结果

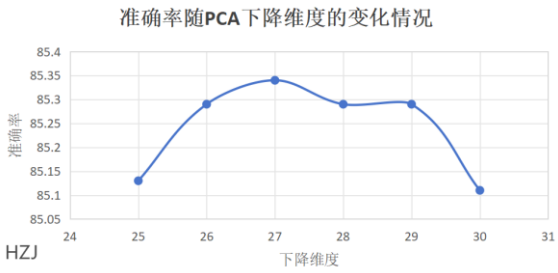
为了探寻最佳的准确率对应的实验参数，本实验选择了影响较大的参数进行了分析，分别是 PCA 下降的维数和人脸检测中 minNeighbors 的大小选择。至于人脸检测的另一个参数，本实验选择了 scaleFactor 为 1.1，代表每次检测时图像尺寸会按照这个比例进行缩小，每次缩小 10%，可以检测到不同尺寸的人脸，这是表现较好的一个取值；其次，对于 KNN 中邻居数量 n_neighbours 的选择，本实验选择常见的 3 作为邻居数。

（1）PCA 下降维数

PCA 下降的维数代表了 PCA 最终所获得特征维度（低维空间维数），是从该特征所有分量中提取到的主特征，代表了数据中最重要的特征，因此选取合适的 PCA 维数对于捕捉人脸特征极其关键。下面设置人脸检测的参数 minNeighbors 为 5，在 10-100，10 为间距进行降维和人脸识别，计算准确率；值得注意的是，在上述实验条件下检测出训练集中人脸的数量为 87（原数 100），测试集中人脸数量为 3861（原数 5194），有些图片识别不出来是因为人脸发生遮挡或者模糊不清。具体结果如下所示：



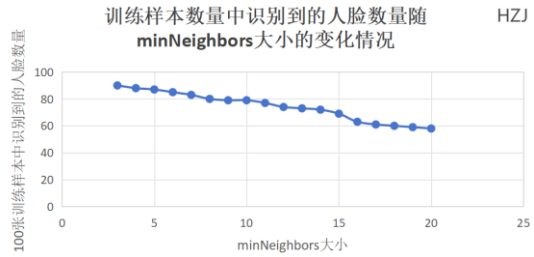
我们可以看到维度在 30 之前成上升趋势，之后成下降趋势，于是在 20-30 之间进一步缩小区域，发现 21-25 仍呈上升趋势（数据不加展示），于是最终将区域锁定在 25-30 之间，具体实验结果如下：



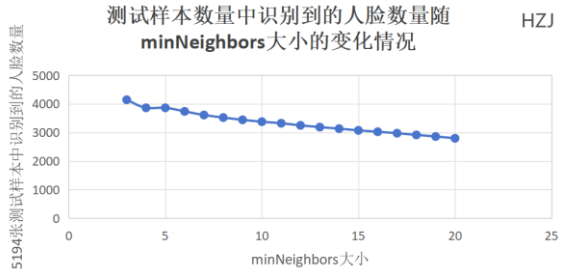
从上述实验结果我们可以看到，当 PCA 下降维数为 27 时准确率最高，为 85.35%，此时 PCA 提取到的人脸特征最能代表训练集中的人脸。

（2）**人脸检测的参数 minNeighbors 的大小**

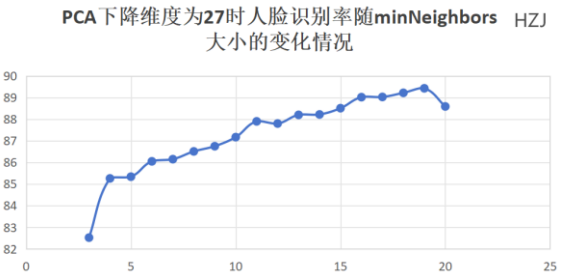
minNeighbors 指定了人脸检测出的每个候选矩形需要保留的邻居数量，即检测到的每个矩形至少需要有多少个邻居矩形（即重叠部分大于某个阈值）才能被认为是有效的人脸。这个参数越大，代表人脸检测越来越严格，如果遮挡超过一定阈值，人脸检测将不会检测到人脸，这意味着检测到的人脸数量越少。选择前面已经找到的最佳 PCA 下降维数 27 时，从 3-20 之间，以 1 为间隔进行实验。具体结果如下：



由实验结果可知随着 minNeighbors 的增大，识别到训练集中的人脸数量越来越少。



由实验结果可知随着 minNeighbors 的增大，识别到测试集中人脸数量也越来越少。



由实验结果可知识别整体缓慢增长，在 minNeighbors 为 20 时基本停滞甚至开始下降。我们猜测准确率上升的大概率是随着 minNeighbors 的增大，条件更加严苛，筛选出的人脸更加清晰，没有特别明显的遮挡，这样使得 PCA 提取到的特征较好，能够识别到更多人

脸，当然也不排除数据集数量下降带来的影响。选择较大的 minNeighbors 值会使得检测到的数据集数量太少，不利于提取全局特征；选择较小的 minNeighbors 值会使得条件太宽松，数据集噪声干扰太大，使得人脸识别准确率下降。因此如何平衡选择较好的 minNeighbors 值进行人脸检测较为重要，根据实际需要进行设置。

四、实验结论：

本次实验通过划分训练和测试视频序列数据集，利用 Haar 特征分类器进行人脸检测，并将检测到训练集的人脸进行 PCA 降维得到投影矩阵，并用检测到的测试集人脸验证 PCA 得到的投影矩阵的有效性吗，并用 KNN 计算准确率作为标准，最终准确率维持在 85%-90%之间，成功验证了 PCA 提取特征的有效性，同时对实验影响较大的参数进行了多次实验和分析，得到较好的参数取值。本次实验原名结束。

五、实验体会：

本次实验进行了视频人脸特征提取实验，由于之前上过机器学习的课，所以对 PCA 还是比较熟悉，但实验较为新颖的还是 Haar 特征分类器，利用 AdaBoost 算法进行训练，最后提取特征并检测出人脸区域。这个对我启发还是挺大，毕竟本人之前做 PCA 实验数据给的基本都是人脸数据，忽略了现实生活中获取的图片人脸不一定是位居图片主体，还需要通过人脸检测和分割得到具体的图片，再进行 PCA 降维特征提取。通过本次实验，回顾和加深了 PCA 的理解，了解了 Haar 特征分类器，收获颇丰。

指导教师批阅意见：

成绩评定：

指导教师签字：沈琳琳、文嘉俊

2024 年 11 月 5 日

备注：