人工智能程序设计期末设计

——基于 PCA+KNN，Eigenfaces（自行设计），CNN 实现的人脸识别

白晋斌

171860607

Table of Contents

一、项目简述

作品名称：基于 PCA+KNN，Eigenfaces（自行设计），CNN 实现的人脸识别对比

小组成员

学号	姓名	院系	手机	邮箱	在项目小组中承担的任务（简写）
171860607	白晋斌	计算机科学与技术系	17551082554	810594956@qq.com	三种方法的人脸识别程序编写与优化

背景和问题

详见第二部分。关于原始项目中所提问题的回答在第七部分。

设计思路

首先对原始数据清理，然后尝试使用例如 PCA+KNN，Eigenfaces（自行设计），CNN 等多种方案实现人脸识别，并对识别效果进行比对。最后展示一些可视化效果。

数据来源

助教所提供链接 https://cswww.essex.ac.uk/mv/allfaces/faces94.html

处理分析

详见第三到六部分。

结论

基于自行设计的 Eigenfaces 模型，最终在测试集上达到了 100%的准确性。启发我们在进行模型训练时，如果数据集较小，可以采用一些基础的、简单的算法，而不是想都不想直接来深度学习。

其他想说的（可选）

图像可以转化为二维数据，似乎音频特征可以转化为一位数据；图像可以基于人脸特征点进行分类，那么音频该基于什么呢？准备暑假做个研究尝试。

二、背景和问题

人脸识别

任务描述：

在 Faces94 数据集上用 python3 实现人脸识别模型，统计在测试集上的整体 识别率（正确识别的张数/测试集总张数*100%）。

数据集： Faces94 数据集简介: 一共 153*20=3060 张，图片大小 200*180。 数据集介绍和下载地址：

https://cswww.essex.ac.uk/mv/allfaces/faces94.html

建议流程：

1） 【数据获取与预处理】可自行划分训练集和测试集，思考如何获得标注信 息（类别可以用什么表示）

以下给出一种划分方法，训练集和测试集各随机取一半：

train_idx, test_idx = train_test_split(range(len(all_imgs)), test_size=0.5, random_state=200)

2） 【问题分析与建模】运用所学知识对人脸识别问题进行建模（用数学公式 表达如何识别出是哪一个人的面部）。

建议以 KNN(最近邻分类)为基础，探索其他更好的方法。

例子：用 KNN 来建模的话，对于图片 X_i 与数据集中其他图片 X_j(j 不等于 i)，判断图片 X_i 属于哪一类可以这样：求使得(X_i - X_j)^2(j 不等于 i)最小的 j，X_i 判别为和这个 X_j 的类别一样。

若你做了一些变换，这里先用 f(.)描述，则：求使得[f(X_i) - f(X_j)]^2(j 不等于 i)最小的 j，X_i 判别为和这个 X_j 的类别一样。你需要具体(或大致)表达出这个 f(.)。

3） 【代码实现与模型训练】若 opencv 自带了你选取的方法的函数，可作为 自己实现的参考。

4） 【模型测试与评价】a) 给一张图片输出其是哪个人 b) 在测试集上的识别 率是多少？若有超参数，调整超参数使识别率尽量高。

5） 【可视化】（选做）若你的模型对人脸进行了什么变换，尝试变回来（人脸 重建）。

6） 【思考总结】你的模型有什么优缺点，什么情况下适用，什么情况下不适用。

提示：

可能会出现内存问题，如何解决？

三、数据处理

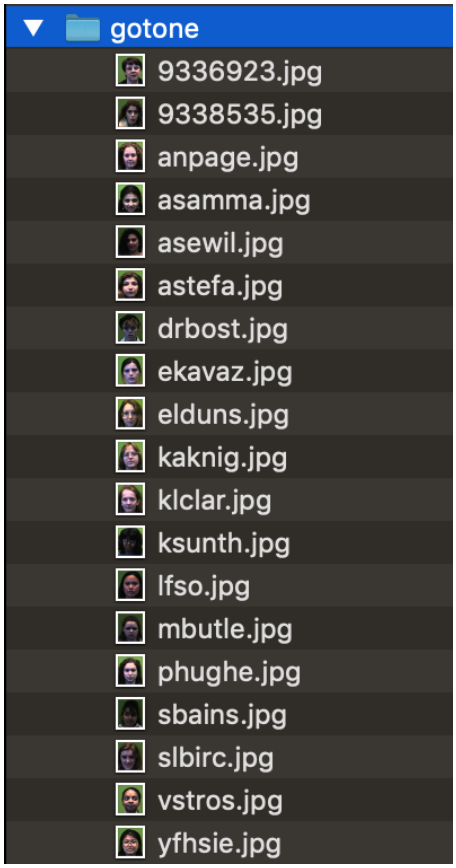
编写如下程序，删除乱七八糟的非 jpg 文件

```
import os
def read_file(path):
```

```
    count=0
    for child_dir in os.listdir(path):
        if child_dir == '.DS_Store':
            os.remove(os.path.join(path, child_dir))
        else:
            child_path = os.path.join(path, child_dir)
            right=0
            this = 0
            for child_file in os.listdir(child_path):

                if child_file == '.DS_Store':
                    os.remove(os.path.join(child_path, child_file))
                elif child_file[-3:]!='jpg':
                    os.remove(os.path.join(child_path, child_file))
                else:
                    count+=1
                    this+=1
                if this==20:
                    right=1
            if right==0:
                print(child_path)

    return count

if __name__ == '__main__':
    print(read_file('data/'))
```

然后我们发现有个文件夹里面不是一个人的照片，乱七八糟的而且有 19 张，这样我们删掉这个文件夹。

```
data/gotone
3059
```



现在我们有 152*20=3040 张图片作为数据集。

三、初步尝试

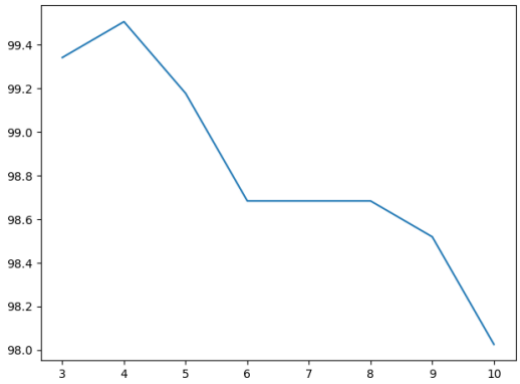
接着我们读取数据，按照 8:2 的比例将数据集划分为训练集和测试集。对图片的处理包括：将 RGB 转化为灰度，将 200*180 铺展成 36000*1 一个维度，并对数据做归一化处理，以便后续 KNN 计算距离。每个图片所在的文件夹名字即为他的标签，最后我们将标签数字化为 0，1，2 ……。

使用加权 KNN 算法，首先求测试数据与所有训练数据的欧氏距离，对其排序。获得经过排序的距离值，再取距离最近的 k 条数据。对前 k 个距离计算权重，并确定它们所在的类别，加权后返回数值最大的对应预测的分类。

$$f(x) = \frac{\sum_{i=1}^{k} D_i W_i}{\sum_{i=1}^{k} W_i}$$

D_i代表近邻 i 与待预测值 x 的距离，Wi 代表其权重，f(x)是预测的数值型结果。每预测一个新样本的所属类别时，都会对整体样本进行遍历，可以看出 kNN 的效率实际上是十分低下的。

如何取 k 值呢？我们尝试了 3-10 多种情况。最终发现，K 值设定在 3 或 4 预测准确性最高。如下图所示。



在实际测试中，我们的预测准确性达 99.67%，仅仅猜错两张图片。但效率很低，预测一张图片平均要一秒钟。

```
face recognization accuracy: 99.67105263157895%
```

分析两张出错的图片，发现竟然是？？！


头歪了？

由此设想，如果我们简单的把图片转置，分类结果将可能会很差。经实验证明，对图像做旋转处理后，KNN 直接分类基本失效。

四、提高效率

鉴于上一步中预测效率过低，这里我们准备采用 PCA 降维对每张图片 36000*1 的数据进行降维。

PCA 的计算原理如下：

1）构建 p*n 阶的变量矩阵 X

2）将 p*n 阶的变量矩阵 X 的每一行（代表一个属性字段）进行标准化

3）求出协方差矩阵 C

4）求出协方差矩阵的特征值及对应的特征向量

5）将特征向量按对应特征值大小从上到下按行排列成矩阵，并且取前 k 列组成矩阵 P

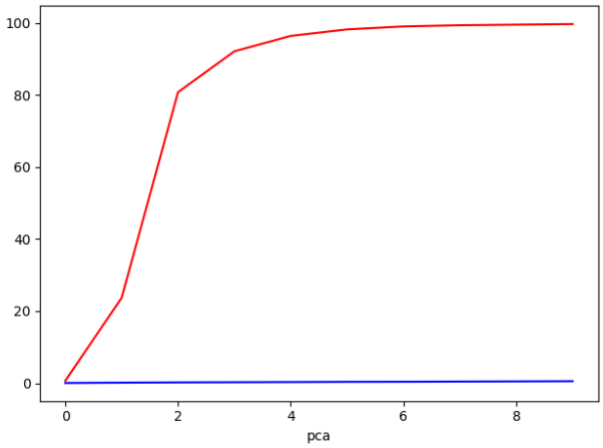
6）Y=XP 即为降维到 k 维后的数据

首先将图片降维成 360 维，降维用时 34.31304407119751 秒，图像识别用时 18.44404101371765 秒，正确率 99.67105263157895%，可以说其实正确率没有降低，还是那两张奇怪的图片。比起之前约十分钟的图像识别，可谓大幅度提升。

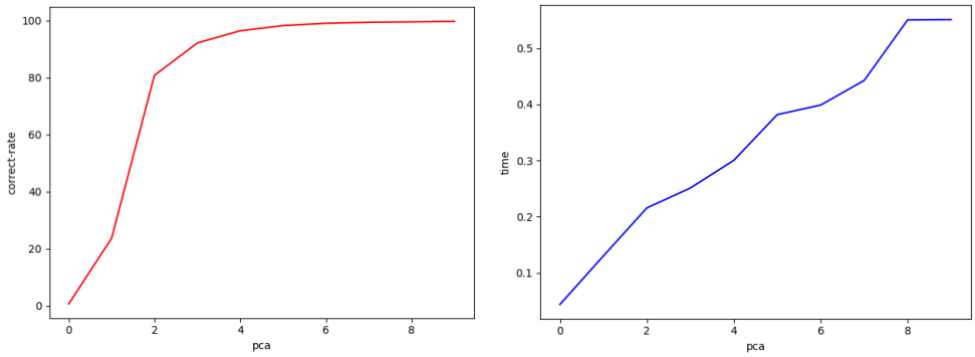
接着我们将图片降维成 36 维，降维用时 15.427454948425293 秒，图像识别用时 1.784952163696289 秒，正确率 99.67105263157895%，可以说其实正确率没有降低，还是那两张奇怪的图片。比起之前近 1 分钟的图像识别，可谓大幅度提升。

接着我们将图片降维成 6 维，降维用时 14.143389225006104 秒，图像识别用时 0.4108858108520508 秒，正确率 98.51973684210526%，正确率略微降低。比起之前 1.7 秒的图像识别，可谓大幅度提升。

对降维到 1-10 维的时间与正确率绘图。



如图，红线代表正确率，可以看出降维在 5 维以上，正确率就接近 100%了。分别绘图：



可知降维到 8-9 维是较为理想的结果。

五、提高正确率

速度是上去了，可能那两张图老是猜错，该怎么办呢？这里我们想到对人脸建模。常见的建模方法有：

（1）主成分分析（PCA）——Eigenfaces（特征脸）

函数：cv2.face.EigenFaceRecognizer_create（）

PCA：低维子空间是使用主元分析找到的，找具有最大方差的哪个轴。

缺点：若变化基于外部（光照），最大方差轴不一定包括鉴别信息，不能实行分类。

（2）线性判别分析（LDA）——Fisherfaces（特征脸）

函数： cv2.face.FisherFaceRecognizer_create()

LDA:线性鉴别的特定类投影方法，目标：实现类内方差最小，类间方差最大。

（3）局部二值模式（LBP）——LocalBinary Patterns Histograms

函数：cv2.face.LBPHFaceRecognizer_create()

PCA 和 LDA 采用整体方法进行人脸辨别，LBP 采用局部特征提取。

这里我们手工实现 Eigenfaces 算法。

Eigenfaces 就是特征脸的意思，是一种从主成分分析（Principal Component Analysis，PCA）中导出的人脸识别和描述技术。特征脸方法的主要思路就是将输入的人脸图像看作一个个矩阵，通过在人脸空间中一组正交向量，并选择最重要的正交向量，作为“主成分”来描述原来的人脸空间。特征脸方法就是将 PCA 方法应用到人脸识别中，将人脸图像看成是原始数据集，使用 PCA 方法对其进行处理和降维，得到“主成分”——即特征脸，然后每个人脸都可以用特征脸的组合进行表示。这种方法的核心思路是认为同一类事物必然存在相同特性（主成分），通过将同一目标（人脸图像）的特性寻在出来，就可以用来区分不同的事物了。

特征脸代码编写思路（先计算特征脸，然后识别人脸）：

将训练集中的 N 个人 reshape 成一行，然后组合在一起形成一个大矩阵 A。若人脸图像大小为 200 * 180，则矩阵 A 的维度是 N *200 * 180；将 N 个人脸在对应的维度求平均，得到一个“平均脸”，如图。

将矩阵 A 中 N 个图像都减去“平均脸”，得到新矩阵 B；

计算 B 的协方差矩阵；

计算协方差矩阵的特征值和特征向量（特征脸）；

将训练集图像和测试集图像都投影到特征向量空间中，再使用聚类方法 KNN 得到里测试集中的每个图像最近的图像，进行分类即可。

写好了，代码时间复杂度太高？算不出结果？经断点分析，发现是我们的协方差矩阵维度太高。

于是我们准备先降维。根据之前的分析，我们将 Eigenfaces 先降维到 10 维。正确性 99.84%，只猜错一个，还是比之前单纯的 PCA+KNN 要好一点。

```
face recognization accuracy: 99.83552631578947%
```

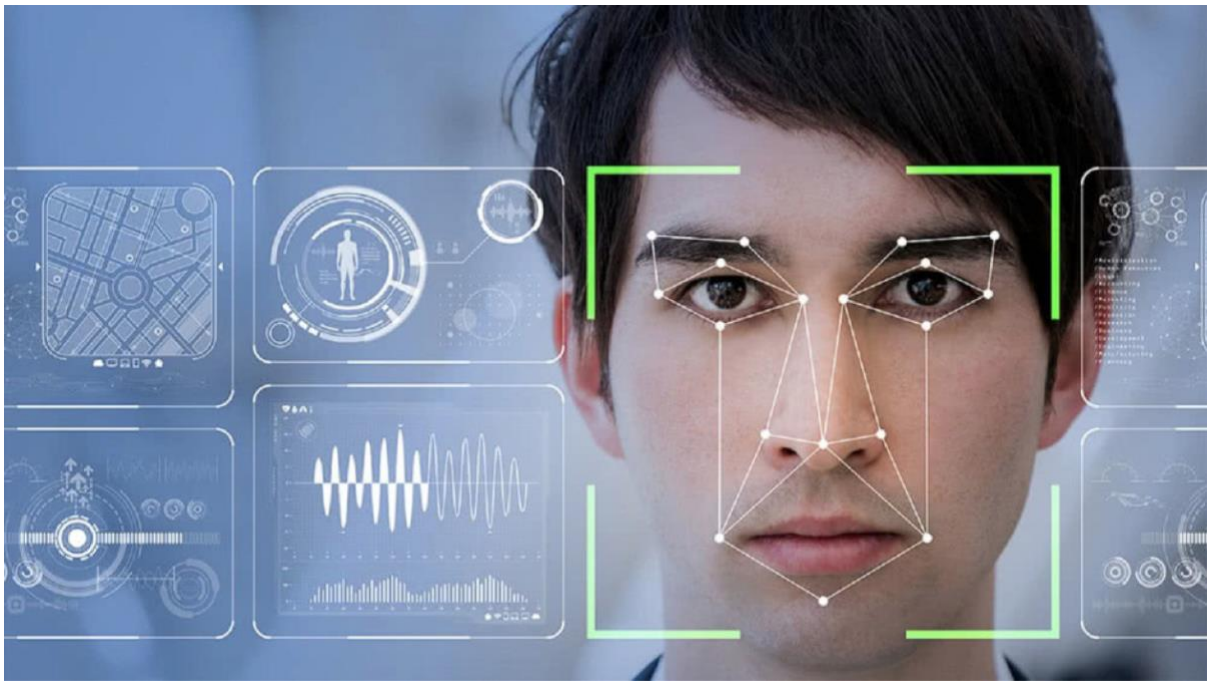
怎么继续优化呢？我们自然的想到了对数据做归一化处理。结果很令人惊喜。识别准确度达 100%。

```
face recognization accuracy: 100.0%
```

接下来说一下自己理解的该算法的局限性。

要让系统准确识别需要保证人脸图像满足：待识别图像中人脸尺寸接近特征脸中人脸的尺寸；待识别人脸图像必须为正面人脸图像。若不满足此条件，识别错误率很高。从 PCA 方法的过程可以看出，特征脸识别的方法是以每张人脸的一个维度（可以看出是矩阵的一列）为单位进行处理的，求得的特征向量（特征脸）中包含训练集每个纬度的绝大部分信息。但是若测试集中人脸尺寸不同，那么与特征脸中维度的也就没法对应起来。

当然对应的改进方案有如老师 PPT 所引用图片，对人脸的 18 个关键点建模，求关键点之间的距离从而获得测试集的分类结果。



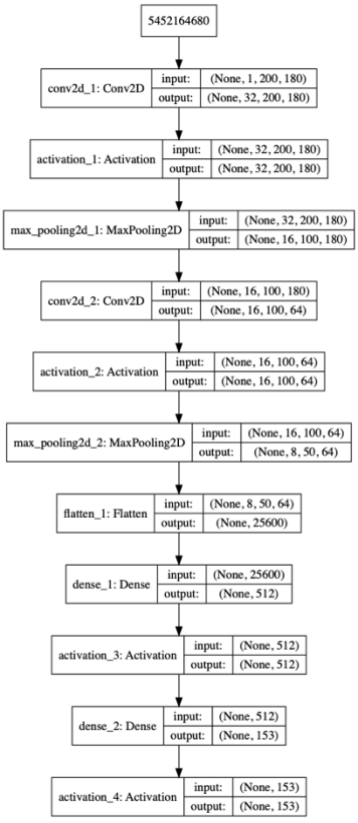
六、深度学习

因本人对 CNN（卷积神经网络）较为熟悉，且 CNN 常用来处理一些图像问题，故这里采用 CNN 对图像集进行分类。

模型较为简单，由卷积层、池化层、卷积层、池化层构成，最后加一个全连接层。详细描述如下图。

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 200, 180)	832
activation_1 (Activation)	(None, 32, 200, 180)	0
max_pooling2d_1 (MaxPooling2	(None, 16, 100, 180)	0
conv2d_2 (Conv2D)	(None, 16, 100, 64)	288064
activation_2 (Activation)	(None, 16, 100, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 8, 50, 64)	0
flatten_1 (Flatten)	(None, 25600)	0
dense_1 (Dense)	(None, 512)	13107712
activation_3 (Activation)	(None, 512)	0
dense_2 (Dense)	(None, 153)	78489
activation_4 (Activation)	(None, 153)	0

Total params: 13,475,097
Trainable params: 13,475,097



多次测试，发现两层卷积池化效果较好，最高准确性可达 99.8%，平均准确性也在 99%以上。然而，耗费大量的计算资源，获得的结果并没有比简单的 Eigenfaces 好很多。这启发我们在进行模型训练时，如果数据集较小，可以采用一些基础的、简单的算法，而不是想都不想直接来深度学习。

```
test loss; 0.02723735732823805
test accuracy: 0.9983660130718954
Model Saved.
```

```
test loss; 0.10660974629713356
test accuracy: 0.9852941176470589
Model Saved.
```

七、可视化

以下可视化展示了一部分我们对图像所做的处理。

上图为原图

进行灰度处理的图

对同一个人的人脸叠加后

对上图做灰度处理

对所有人的人脸叠加

对上图的 200*180 的值取整

七、问题回答

1） 【数据获取与预处理】可自行划分训练集和测试集，思考如何获得标注信息

这里数据集的划分主要采取两种方案，一种是全部读取后根据传入的参数随机将数据分为两部分，另一种是在读取文件是，将最后读取的一定比例图片作为测试集。经测试，二者效果相当。

预处理主要包含降维、归一化、梯度处理等。

2） 【问题分析与建模】运用所学知识对人脸识别问题进行建模（用数学公式表达如何识别出是哪一个人的面部）。

具体的公式与计算思路详见第二到第六部分。

3） 【代码实现与模型训练】若 opencv 自带了你选取的方法的函数，可作为自己实现的参考。

对比自行实现的 Fisherfaces 与 cv2 自带的 Fisherfaces，发现在模型精确度上无明显差别，但在运行效率上，cv2 自带函数所耗时间是自己所编写函数的 1/3 到 1/5。

4） 【模型测试与评价】a) 给一张图片输出其是哪个人 b) 在测试集上的识别率是多少？若有超参数，调整超参数使识别率尽量高。

经调试，KNN 取 k=3 左右较为合适，PCA 降维取 k=9 左右较为合适。在各种优化之后，模型的精确性从 98%最终成功达到 100%。

5） 【可视化】（选做）若你的模型对人脸进行了什么变换，尝试变回来（人脸 重建）。

该部分详见第七部分。

6） 【思考总结】你的模型有什么优缺点，什么情况下适用，什么情况下不适用。

我的模型需要人脸图像满足：待识别图像中人脸尺寸接近特征脸中人脸的尺寸；待识别人脸图像必须为正面人脸图像。若不满足此条件，识别错误率很高。从PCA 方法的过程可以看出，特征脸识别的方法是以每张人脸的一个维度（可以看出是矩阵的一列）为单位进行处理的，求得的特征向量（特征脸）中包含训练集每个纬度的绝大部分信息。但是若测试集中人脸尺寸不同，那么与特征脸中维度的也就没法对应起来。

所以对于标准的证件照识别与分类效果很好（速度还很快），但对于街拍、视频抓取效果则可能不会很好。

7） 可能会出现内存问题，如何解决？

并未出现该问题，不过认为可以在读取文本时采用灰度读取，读取后即对数据做降维处理。