机器学习期末课程报告

PCA 在人脸识别中的应用

【专业】计算机科学与技术　　【学号】22336259　　【姓名】谢宇桐

一、研究问题的背景和动机

背景

人脸识别技术是一种重要的生物识别技术，广泛应用于安全认证、监控系统、人机交互等领域。在实际应用中，人脸识别系统需要处理高维数据（如人脸图像），这些数据通常包含大量的冗余信息和噪声。此外，人脸图像的复杂性（如光照变化、姿态变化、表情变化等）给识别任务带来了巨大挑战。

动机

主成分分析（PCA）是一种经典的线性降维技术，能够有效地将高维数据投影到低维空间，同时保留数据的主要结构信息。在人脸识别领域，PCA 被广泛应用于特征提取，通过将人脸图像转换为一组“特征脸”（Eigenfaces），可以显著降低计算复杂度，并提高识别性能。本研究旨在通过实现基于 PCA 的人脸识别系统，验证其在不同降维维度下的性能表现，并探讨其在实际应用中的可行性。

二、简要概述当前解决该问题的主要方法

传统方法

1. **模板匹配**：直接将输入图像与数据库中的模板图像进行相似性比较。这种方法对光照和姿态变化敏感，且计算复杂度较高。
2. **几何特征方法**：提取人脸的关键特征点（如眼睛、鼻子、嘴巴的位置和距离），并基于这些特征进行识别。这种方法对姿态和表情变化较为鲁棒，但对光照变化敏感。

基于 PCA 的方法

1. **特征脸方法（Eigenfaces）**：通过 PCA 将人脸图像降维到低维空间，提取一组“特征脸”作为人脸的特征表示。该方法在光照和姿态变化较小的情况下表现出色。
2. **改进的 PCA 方法**：结合其他技术（如线性判别分析 LDA、核 PCA 等）进一步提升识别性能。

深度学习方法

1. **卷积神经网络（CNN）**：通过学习人脸图像的深层特征，CNN 在人脸识别任务中取得了显著的性能提升。然而，CNN 需要大量的训练数据和计算资源。
2. **生成对抗网络（GAN）**：用于生成高质量的人脸图像，以增强数据集的多样性和鲁棒性。

三、详细阐述模型、算法或方法

首先介绍一下PCA的定义和原理：

主成分分析（Principal Component Analysis，简称PCA）是一种用于数据降维的技术，旨在通过线性变换将原始数据映射到一个新的坐标系中。在这个新的坐标系中，数据的方差被最大化，这意味着我们将保留尽可能多的原始数据信息。PCA的关键思想是通过找到数据中的主成分，将数据在这些主成分上进行投影，从而实现数据的降维。

本研究采用基于 PCA 的特征脸方法（Eigenfaces）进行人脸识别。具体步骤如下：

1. **数据预处理**
   - 收集人脸图像数据集，并对图像进行预处理，包括人脸检测、对齐和归一化。
   - 将人脸图像转换为灰度图像，并将其展平为一维向量。假设每张图像的大小为 112×92，则展平后的向量长度为 10304。
   - 对图像数据进行矢量化处理，将每张图片转换为一个 1×10304 的向量。
2. **PCA 分析**
   - 构建数据矩阵 X，其中每一列是一个人脸图像向量。
   - 对数据矩阵进行中心化处理，即减去每列的均值向量 μ。
   - 计算协方差矩阵 S=N1XTX，并对协方差矩阵进行特征分解，提取前 r 个主成分（特征脸）。
   - 对特征向量进行归一化处理，确保其单位范数。
3. **特征提取**
   - 对每张人脸图像，计算其在特征脸空间中的投影，即特征向量：

     yi=UrT(xi−μ)

     其中，Ur 是包含前 r 个主成分的矩阵。
4. **人脸识别**
   - 对于输入的人脸图像，提取其特征向量，并与数据库中的特征向量进行比较。
   - 采用最近邻分类器（如欧氏距离）进行分类，识别出最相似的人脸。

下面，我们结合代码进行详细解释：

1. ## 图像矢量化（img2vector）

   在人脸识别任务中，图像数据需要被转换为适合数值计算的形式。`img2vector` 函数的作用是将二维的灰度图像转换为一维向量。在本次实验中，我们使用ORL官方数据集，该数据集表示的是一共有40个人的人脸图像，其中每一个人有10张人脸图像。

```python
def img2vector(image):
    """
    将图像文件转换为一维向量。
    参数:image (str): 图像文件的路径。
    返回:imgVector (numpy.ndarray): 一维向量形式的图像数据。
    """
    # 使用 OpenCV 读取图像，0 表示以灰度模式读取
    img = cv2.imread(image, 0)

    # 将图像矩阵直接展平为一维向量
    imgVector = img.flatten().reshape(1, -1)  # 使用 flatten 展平，然后重新塑形为
(1, rows*cols)

    return imgVector
```

2. 数据加载与划分（`load_orl`）

load_orl 函数的作用是从 ORL 数据集中加载人脸图像，并将其划分为训练集和测试集。

我们使用的图片集的文件夹层级结构如下：

```
D:\homework\AI\machine\PCA\orl_faces
├── s1
|   ├── 1.pgm
|   ├── 2.pgm
|   ├── 3.pgm
|   ├── 4.pgm
|   ├── 5.pgm
|   ├── 6.pgm
|   ├── 7.pgm
|   ├── 8.pgm
|   ├── 9.pgm
|   └── 10.pgm
├── s2
|   ├── 1.pgm
|   ├── 2.pgm
|   ├── 3.pgm
|   ...
|   ├── 8.pgm
|   ├── 9.pgm
|   └── 10.pgm
...
└── s40
    ├── 1.pgm
    ├── 2.pgm
    ├── 3.pgm
    ├── 4.pgm
    ├── ...
    ├── 9.pgm
    └── 10.pgm
```

```python
# 读入人脸库,每个人随机选择k张作为训练集,其余构成测试集
def load_orl(k):
```

```
    """
    参数:k (int): 每个人选择的训练样本数量。
    返回:
        train_face (numpy.ndarray): 训练集图像数据。
        train_label (numpy.ndarray): 训练集标签。
        test_face (numpy.ndarray): 测试集图像数据。
        test_label (numpy.ndarray): 测试集标签。
    """
    orlpath = "./orl_faces"  # 数据集路径
    num_people = 40  # 总人数
    num_images_per_person = 10  # 每个人的图像数量
    image_size = 112 * 92  # 图像尺寸

    # 初始化训练集和测试集
    train_face = np.zeros((num_people * k, image_size))
    train_label = np.zeros(num_people * k, dtype=int)
    test_face = np.zeros((num_people * (num_images_per_person - k),
image_size))
    test_label = np.zeros(num_people * (num_images_per_person - k),
dtype=int)

    for i in range(num_people):  # 遍历每个人
        people_num = i + 1
        image_indices = list(range(1, num_images_per_person + 1))  # 图像编号
从1到10
        random.shuffle(image_indices)  # 随机打乱图像编号

        # 分配训练集和测试集
        train_indices = image_indices[:k]  # 前 k 张作为训练集
        test_indices = image_indices[k:]  # 剩余的作为测试集

        for j, idx in enumerate(train_indices):  # 遍历训练集图像
            image_path = os.path.join(orlpath, f"s{people_num}", f"
{idx}.pgm")
            img_vector = img2vector(image_path)  # 将图像转换为一维向量
            train_face[i * k + j] = img_vector
            train_label[i * k + j] = people_num

        for j, idx in enumerate(test_indices):  # 遍历测试集图像
            image_path = os.path.join(orlpath, f"s{people_num}", f"
{idx}.pgm")
            img_vector = img2vector(image_path)  # 将图像转换为一维向量
            test_face[i * (num_images_per_person - k) + j] = img_vector
            test_label[i * (num_images_per_person - k) + j] = people_num

    return train_face, train_label, test_face, test_label
```

3. PCA 算法实现

PCA 的目标是将高维数据投影到低维空间，同时保留数据的主要结构信息。输入数据 data 是人脸图像的矢量化数据，r 是降维后的维度。

```
def PCA(data, r):
    data = np.float32(np.mat(data))
    rows, cols = np.shape(data)

    # 数据标准化
    data_mean = np.mean(data, 0)  # 对列求平均值
    A = data - np.tile(data_mean, (rows, 1))  # 去中心化

    C = A * A.T  # 计算协方差矩阵
    D, V = np.linalg.eig(C)  # 求协方差矩阵的特征值和特征向量
    V_r = V[:, 0:r]  # 取前r个特征向量
    V_r = A.T * V_r  # 将小矩阵特征向量转换为大矩阵特征向量

    # 归一化特征向量
    for i in range(r):
        V_r[:, i] = V_r[:, i] / np.linalg.norm(V_r[:, i])

    # 计算降维后的数据
    final_data = A * V_r

    return final_data, data_mean, V_r
```

4. 人脸识别

人脸识别部分的核心是利用降维后的特征进行分类。

```
def face_rec():
    dims = range(10, 41, 10)  # 降维维度
    accuracy_list = []
    x_value_list = []

    for r in dims:
        print(f"\n降到{r}维时")
        x_value = []
        y_value = []
        for k in range(1, 10):
            train_face, train_label, test_face, test_label = load_orl(k)  #
得到数据集

            # 利用PCA算法进行训练
            data_train_new, data_mean, V_r = PCA(train_face, r)
            num_train = data_train_new.shape[0]  # 训练脸总数
            num_test = test_face.shape[0]  # 测试脸总数
            temp_face = test_face - np.tile(data_mean, (num_test, 1))
            data_test_new = temp_face * V_r  # 得到测试脸在特征向量下的数据
            data_test_new = np.array(data_test_new)  # mat change to array
            data_train_new = np.array(data_train_new)
```

```
            # 测试准确度
            true_num = 0
            for i in range(num_test):
                testFace = data_test_new[i, :]
                diffMat = data_train_new - np.tile(testFace, (num_train, 1))
 # 训练数据与测试脸之间距离
                sqDiffMat = diffMat ** 2
                sqDistances = sqDiffMat.sum(axis=1)  # 按行求和
                sortedDistIndicies = sqDistances.argsort()  # 对向量从小到大排
序，使用的是索引值,得到一个向量
                indexMin = sortedDistIndicies[0]  # 距离最近的索引
                if train_label[indexMin] == test_label[i]:
                    true_num += 1

            accuracy = true_num / num_test
            x_value.append(k)
            y_value.append(round(accuracy, 2))

            print(f'每人选择{k}张照片进行训练，准确率为: {accuracy * 100:.2f}%')
            # ...结果可视化部分
```

5. 结果可视化

绘制不同降维维度和训练样本数量下的识别准确率曲线。

```
def plot_accuracy(x_value, y_value, r):
    plt.plot(x_value, y_value, marker="o", markerfacecolor="red")
    for a, b in zip(x_value, y_value):
        plt.text(a, b, (a, b), ha='center', va='bottom', fontsize=10)
    plt.title(f"降到{r}维时识别准确率", fontsize=14)
    plt.xlabel("k值", fontsize=14)
    plt.ylabel("准确率", fontsize=14)
    plt.show()

# 人脸识别
def face_rec():
    dims = range(10, 41, 10)  # 降维维度
    accuracy_list = []
    x_value_list = []

    for r in dims:
        print(f"\n降到{r}维时")
        x_value = []
        y_value = []

        # ...人脸识别

        # 绘图
        plot_accuracy(x_value, y_value, r)
        accuracy_list.append(y_value)
        x_value_list = x_value  # 所有维度的x值是一样的

    # 各维度下准确度比较
    lines = []
```

```
    labels = []
    for i, r in enumerate(dims):
        line, = plt.plot(x_value_list, accuracy_list[i], marker="o",
markerfacecolor="pink")
        lines.append(line)
        labels.append(f"降到{r}维")

    plt.legend(lines, labels, loc=4)
    plt.title("各维度识别准确率比较", fontsize=14)
    plt.xlabel("k值", fontsize=14)
    plt.ylabel("准确率", fontsize=14)
    plt.show()
```

四、实验结果与分析

实验设置

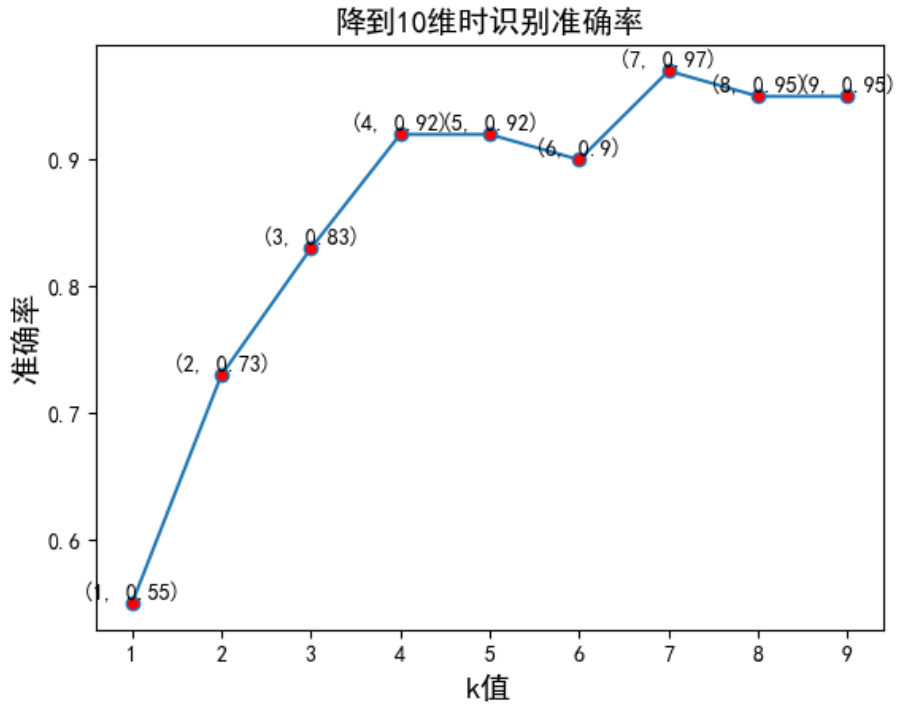
- **数据集**：使用 ORL 人脸数据集，包含 40 个人的 400 张人脸图像（每人 10 张）。大概如下：

- **评价指标**：使用识别率（Recognition Rate）作为评价指标。

实验结果

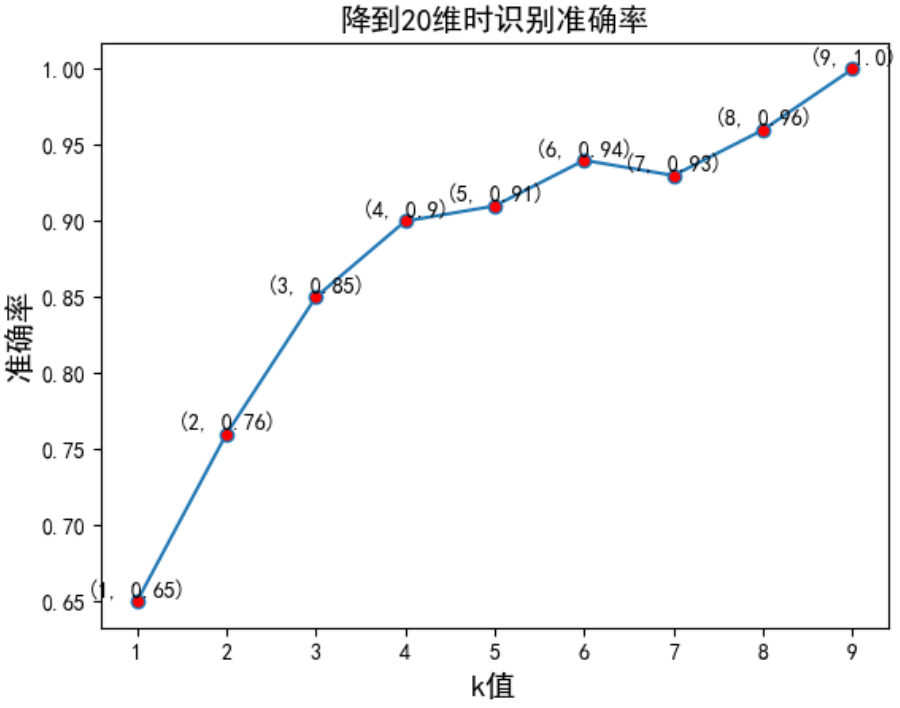
1. **训练样本的影响**
    - 实验分别在降维到 10、20、30 和 40 维时进行测试，结果如下：
        - **降维到 10 维**：当每个人选择 1 张照片进行训练时，识别率约为 55%；选择 5 张照片时，识别率达到 91%；而选择9张照片时，训练可以达到95%。

```
降到10维时
每人选择1张照片进行训练，准确率为: 55.2778%
每人选择2张照片进行训练，准确率为: 72.8125%
每人选择3张照片进行训练，准确率为: 83.2143%
每人选择4张照片进行训练，准确率为: 91.6667%
每人选择5张照片进行训练，准确率为: 91.5000%
每人选择6张照片进行训练，准确率为: 90.0000%
每人选择7张照片进行训练，准确率为: 97.5000%
每人选择8张照片进行训练，准确率为: 95.0000%
每人选择9张照片进行训练，准确率为: 95.0000%
```



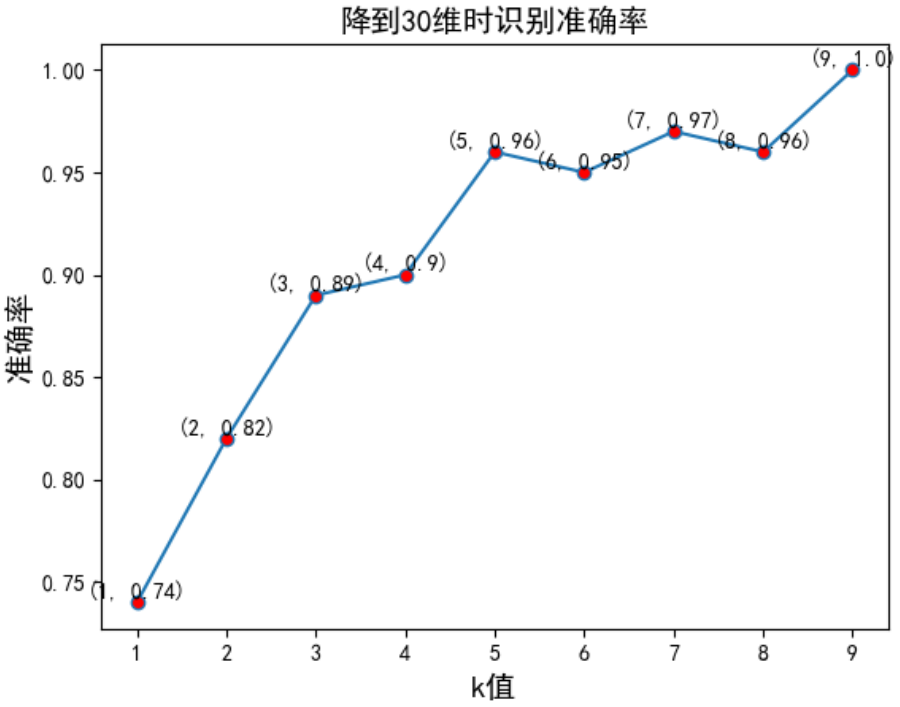
- **降维到 20 维**：当每个人选择 1 张照片进行训练时，识别率约为 65%；选择 5 张照片时，识别率达到 90%；选择9张照片时，识别率可以达到100%。

```
降到20维时
每人选择1张照片进行训练，准确率为：65.2778%
每人选择2张照片进行训练，准确率为：75.6250%
每人选择3张照片进行训练，准确率为：84.6429%
每人选择4张照片进行训练，准确率为：90.4167%
每人选择5张照片进行训练，准确率为：90.5000%
每人选择6张照片进行训练，准确率为：93.7500%
每人选择7张照片进行训练，准确率为：93.3333%
每人选择8张照片进行训练，准确率为：96.2500%
每人选择9张照片进行训练，准确率为：100.0000%
```



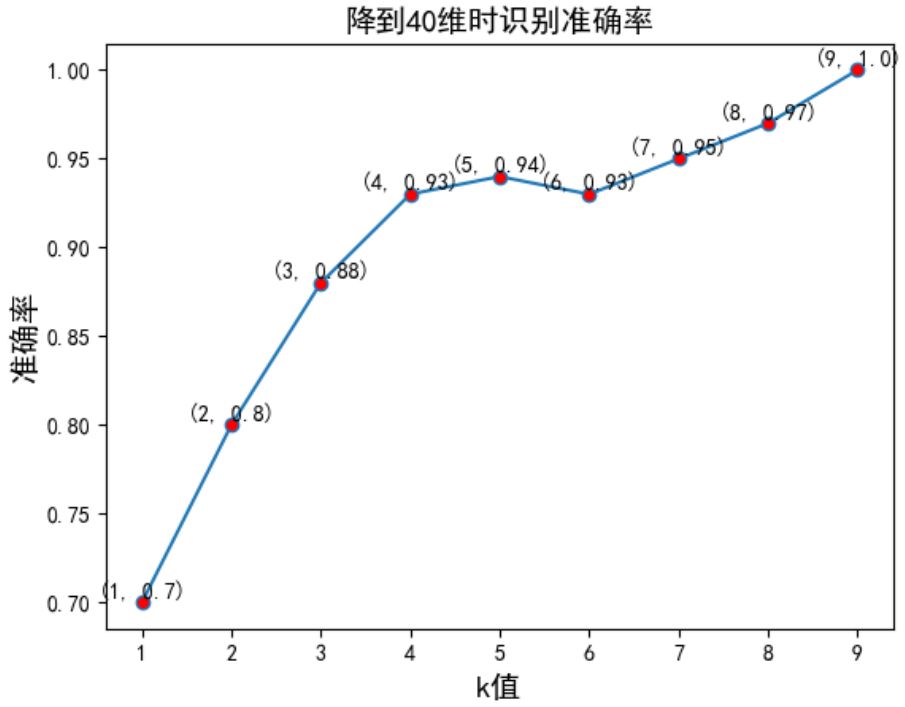
- **降维到 30 维**：当每个人选择 1 张照片进行训练时，识别率为 74%；选择 5 张照片时，识别率达到 96%；选择9张照片时，识别率达到100%。

```
降到30维时
每人选择1张照片进行训练，准确率为: 74.4444%
每人选择2张照片进行训练，准确率为: 81.5625%
每人选择3张照片进行训练，准确率为: 89.2857%
每人选择4张照片进行训练，准确率为: 89.5833%
每人选择5张照片进行训练，准确率为: 96.0000%
每人选择6张照片进行训练，准确率为: 95.0000%
每人选择7张照片进行训练，准确率为: 96.6667%
每人选择8张照片进行训练，准确率为: 96.2500%
每人选择9张照片进行训练，准确率为: 100.0000%
```



- **降维到 40 维**：当每个人选择 1 张照片进行训练时，识别率为 70%；选择 5 张照片时，识别率达到 94%；选择 10 张照片时。准确率可以达到100%

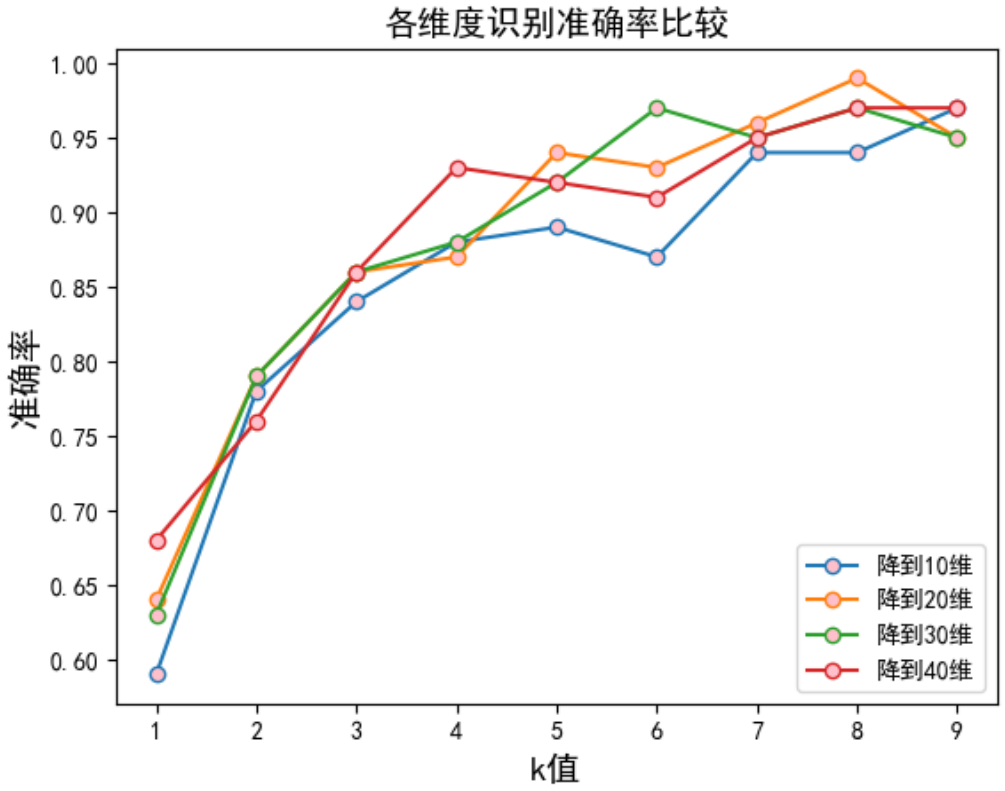
```
降到40维时
每人选择1张照片进行训练，准确率为: 70.0000%
每人选择2张照片进行训练，准确率为: 80.0000%
每人选择3张照片进行训练，准确率为: 87.8571%
每人选择4张照片进行训练，准确率为: 92.5000%
每人选择5张照片进行训练，准确率为: 94.5000%
每人选择6张照片进行训练，准确率为: 92.5000%
每人选择7张照片进行训练，准确率为: 95.0000%
每人选择8张照片进行训练，准确率为: 97.5000%
每人选择9张照片进行训练，准确率为: 100.0000%
```

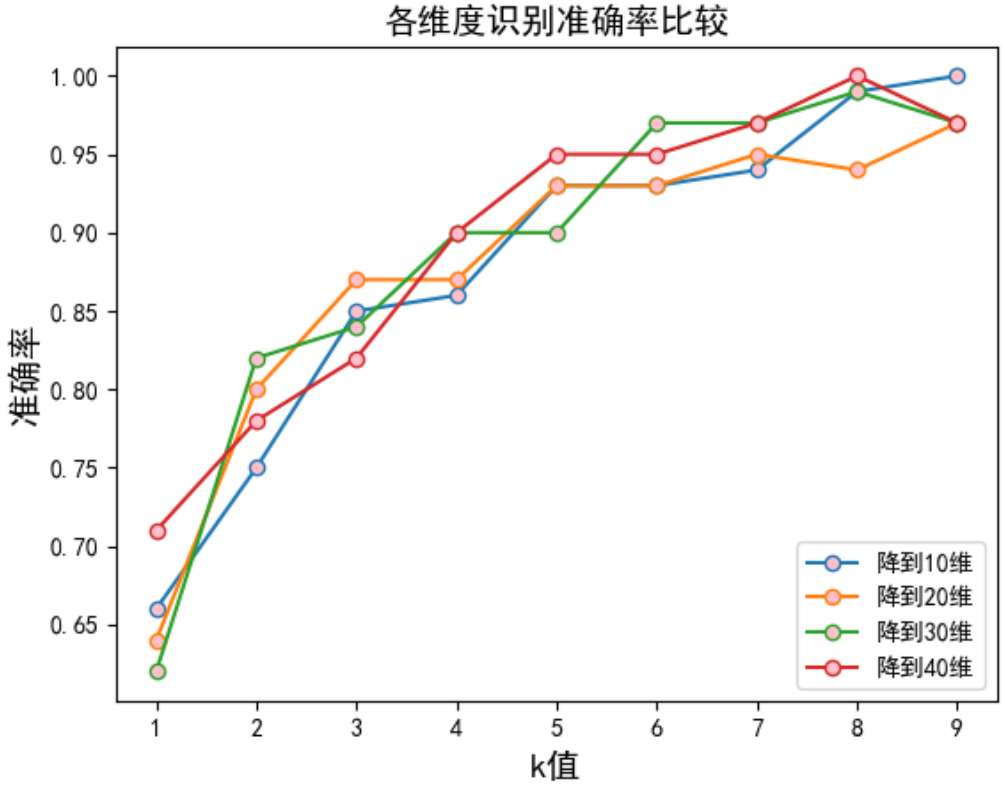


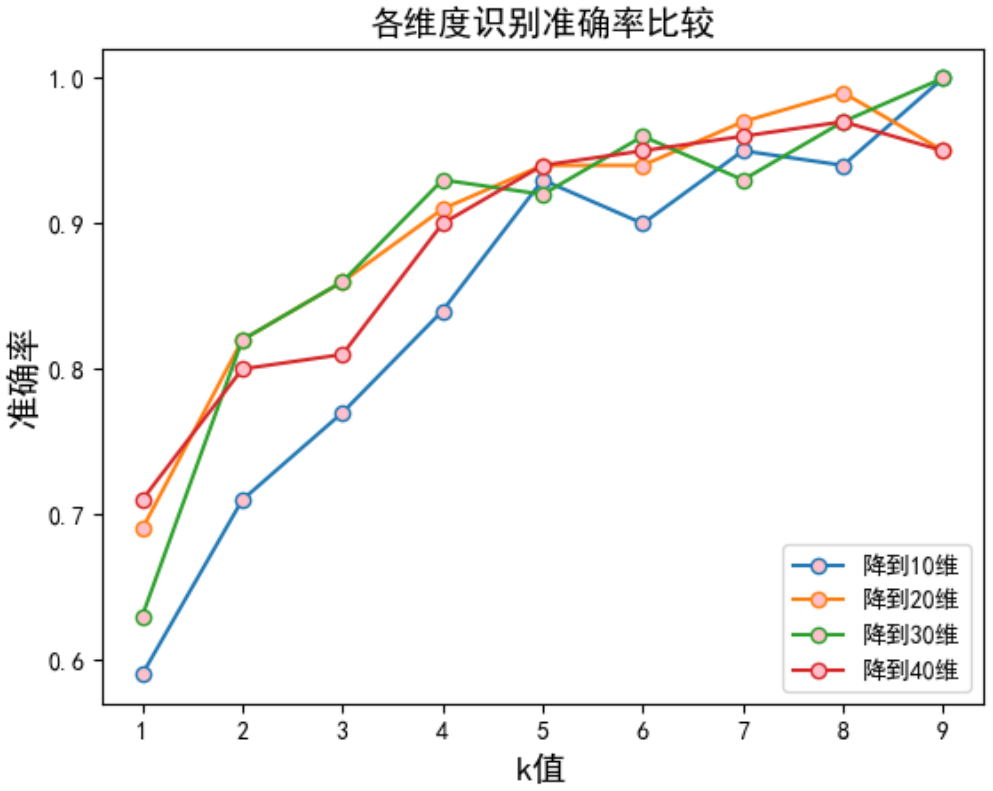
我们可以从曲线中看到，在任何维度下，随着训练样本数量的增加，**模型的识别准确率都是总体升高的趋势**，甚至可以达到100%的正确率。但它并不是单调增加的，这说明了训练时的偶然性。

2. 降低维度的影响

我画了各维度识别准确率进行比较，并多次实验：


各维度识别准确率比较
准确率
k值
降到10维
降到20维
降到30维
降到40维


各维度识别准确率比较
准确率
k值
降到10维
降到20维
降到30维
降到40维



从曲线图我们可以看出，随着降维维度的增加，识别准确率总体上有所提高，且曲线会变得更加平滑。但不是任何时候都会随着维度的降低而增加准确率的。比如在第三张图片中，降到20维度就明显要比降到40维度的准确率要普遍高。总体来说降到20-30维度的准确率会最高。这可能是因为过高的维度保留了一些不重要的信息，导致模型的泛化能力下降。

五、结论

本次实验是基于 PCA 的人脸分析方法，通过提取“特征脸”作为人脸的特征表示，实现了高效的人脸识别。实验结果表明，PCA 方法在 ORL 数据集上取得了较高的识别率，且计算效率较高。然而，PCA 方法对非线性变化的鲁棒性有限。未来可以继续探索结合非线性方法或深度学习技术，以进一步提升人脸识别的性能和鲁棒性。

参考文献

1. Turk, M., & Pentland, A. (1991). Face Recognition Using Eigenfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
2. Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.