雲南大學

《2023 模式识别》

期末实验报告

题　　目：	人脸识别综述及MATLAB实现
上课时间：	2023春季学期
授课教师：	柴晶
姓　　名：	李昂
学　　号：	20201060287
日　　期：	2023年5月30日

人脸识别综述及MATLAB实现

李昂 20201060287

云南大学信息学院 智能科学与技术

摘　要： 在信息化时代，人脸识别技术在各领域得到广泛应用。本文通过介绍人脸识别相关概念、实现流程和实现方法,同时通过Matlab搭建简单的分类器和神经网络,复现人脸识别的经典算法,比较分类的正确率并分析误差原因,加深对模式识别技术的理解。

关键词： 人脸识别；主成分分析；线性判别分析；支持向量机；K最近邻算法；尺度不变特征变换；卷积神经网络；

1　人脸识别综述

学习人脸识别技术，不仅要学习某种方法的实现原理并能够复现相关代码；更要了解人脸识别技术的提出、发展阶段和完成任务的完备操作流程，这样才能从本质理解各种方法的不同，并提出优化方案。基于上述观点，我完成了人脸识别综述部分。该部分基于英国赫特福德大学与 GBG Plc 联合发表的Face Recognition: From Traditional to Deep Learning Methods[1] 对人脸识别技术进行全面的梳理和总结。

1.1　概要

1.1.1　定义

人脸识别是指能够识别或验证图像或视频中主体身份的技术。

1.1.2　优势及应用场景

人脸识别比其他生物识别方式更有吸引力的因素之一是其非侵入性。现代人脸识别系统只要求用户在摄像头的视野范围内。这使得人脸识别成为最方便的生物识别方式。这也意味着人脸识别的潜在应用范围更广，可以部署在用户不需要与系统合作的环境中，如监控系统中。人脸识别的其他常见应用包括访问控制、欺诈检测、身份验证和社交媒体等。

1.1.3　影响因素

现实世界干扰因素众多，在自然状态下获取的图像（这些类型的人脸图像通常被称为野生人脸）常含有众多干扰项，影响人脸识别的特征判断。

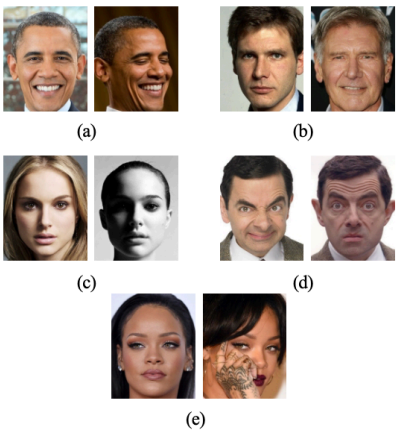

图 1 典型人脸的影响因素: (a)头部姿势 (b)年龄 (c)光照度 (d)面部表情 (e)遮挡

由于人脸图像在现实世界中的高变异性，人脸识别是部署在无约束环境中最具挑战性的生物识别方式之一。其中一些变化包括头部姿势、老化、遮挡、光照条件和面部表情等。

1.2 人脸识别构建流程

野生人脸易受多种因素影响，为了获取足够的信息进行人脸识别，人脸识别的构建通常包含以下基本模块：

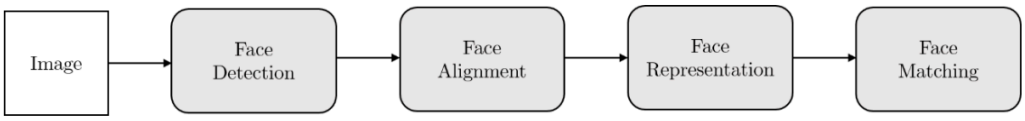

图 2 人脸识别构建模块

- **人脸检测**:人脸检测器找到图像中人脸的位置，并返回每个人脸的边界框的坐标。
- **脸部对齐**: 脸部对齐的目标是使用一组位于图像固定位置的参考点，以相同的方式缩放和裁剪脸部图像。这个过程通常需要使用一个地标检测器找到一组面部地标，找到适合参考点的最佳仿生变换。
- **脸部表示**: 在人脸表示阶段，人脸图像的像素值被转换为一个紧凑的、有辨识度的特征向量，也被称为模板。理想情况下，同一主体的所有面孔都应该映射为类似的特征向量。
- **脸部匹配**: 在人脸匹配模块中，两个模板被比较以产生一个相似度分数，表明它们属于同一主体的可能性。

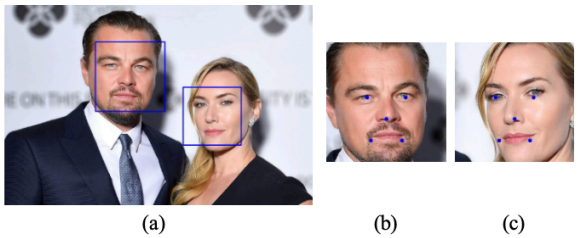

图 3 (a)人脸检测 (b)(c)脸部对齐

以上模块展示了从捕捉野生人脸到人脸识别预处理的过程，这是一个相对复杂的流程，相比本次实验直接通过开源数据集完成，真实情况要复杂的多。

2 MATLAB实现

本次实验使用MATLAB R2022a(Mac Version)，采用ORL数据集。

由于ORL数据集已经进行了完备的标注和预处理，我们得以跳过人脸检测和脸部对齐两个步骤，直接通过不同方法对脸部表示的特征向量进行处理，即可完成人脸识别任务。

以下是具体实验要求：

实验要求
实验时，需要首先载入ORL_trainset.mat文件，根据训练集样本及其类别标签来训练分类器； 然后载入ORL_testset.mat文件，将测试集样本代入已训练好的分类器，得到分类器在测试集上的预测输出； 最后载入ORL_testlabel.mat文件，将分类器的预测输出与测试集样本的类别标签(真实标签)进行对比，计算分类器在测试集上的分类精度，并输出该分类精度。 本实验的部分主程序已经在demo_FaceRecognation.m文件中给出，用来计算分类精度的子程序也已经在calculate_accuracy.m文件中给出，请补足主程序demo_FaceRecognation.m文件中的空缺部分，完成实验。

表 1 实验要求

以下是关于数据集和Demo代码的说明：

数据集和Demo代码的详细说明	
数据集已被预先划分为训练集和测试集	训练集中每类目标有6个样本，共240个样本 测试集中每类目标有4个样本，共160个样本
实验时，需要载入三个.m文件	ORL_trainset.mat文件存储的是训练集样本及其类别标签 ORL_testset.mat文件存储的是测试集样本 ORL_testlabel.mat文件存储的是测试集样本的类别标签

表 2 数据集和Demo代码的详细说明

下面通过几种不同的方式，完成人脸识别任务。

2.1 基于几何的实现方式

2.1.1 原理

基于几何的方法是人脸识别中常用的一种方法，它主要通过分析和比较人脸的几何结构和形状进行识别。通过检测和定位人脸上的关键点（如眼睛、鼻子、嘴巴等）来描述人脸的几何结构。使用边缘检测算法（如Canny边缘检测）来检测人脸图像中的轮廓线，并进一步提取和表示人脸的几何特征。[2]

结合上学期的《图像处理与计算机视觉》课程，基于几何的人脸识别完全依赖于图像处理，通过边缘检测，腐蚀等操作，检测人脸的轮廓，对比后显示结果。本课程关注如何构造合适的分类器完成分类，实现人脸识别功能。基于纯几何的方法不在本课程讨论范围之内，再此不做展开。

2.2 基于整体性的实现方式

> 基于整体特征（holistic-based）的人脸识别方法是指直接使用原始的人脸图像作为输入，并将整个人脸作为一个整体来提取特征进行识别。它不对人脸进行分割或局部特征提取，而是将人脸图像作为一个整体进行,使用支持向量机，K最近邻算法等机器学习的方式进行分类。[2]

整体性方法类似于蛮力法，不提取特征，通过构造分类器完成分类。下面介绍 SVM，KNN的基本原理，并构造分类器，完成人脸识别任务。

2.2.1 基本原理

2.2.1.1 SVM（支持向量机）

> 支持向量机（Support Vector Machine，SVM）是一种常用的监督学习方法，用于二分类和多分类问题。SVM的基本原理是通过构建一个最优的超平面来实现样本的分类，同时最大化分类器与样本之间的间隔，以提高分类的鲁棒性。[2]

支持向量机（Support Vector Machine, SVM）是一种常用的分类方法，在进行分类时通过寻求间隔最大的超平面来进行分类。其数学推导过程较为复杂，简略概括如下：

SVM的目标是要找到最大的间隔（Marginal），其中间隔定义为超平面到最近的训练样本点的距离。

将求间隔转化成求最小化$\frac{1}{2}||\mathbf{w}||^2$，以及满足约束条件$\forall i, y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1$的问题。其中$y_i$表示标签，$\mathbf{x}_i$表示特征向量，$\mathbf{w}^T\mathbf{x}_i+b$表示超平面。

为了处理不可分情况（存在样本分类错误的情况），可以引入软间隔。即允许某些样本分类错误，但是会产生一个惩罚项来弥补错误的损失。最终变成了一个带惩罚项的最小化问题，可以通过拉格朗日对偶性来进行求解。

最终的分类决策函数为：$f(x)=sign(\sum_{i=1}^N \alpha_i y_i K(x,x_i)+b)$，其中$K(x,x_i)$表示核函数。

2.2.1.2 KNN（K最近邻）

> K最近邻（K-Nearest Neighbors，KNN）是一种常用的非参数化监督学习方法，用于分类和回归问题。KNN的基本原理是根据样本之间的距离度量，通过投票或平均值来确定新样本的类别或预测值。[2]

分类问题的KNN算法可以表示为以下步骤：

1. 计算测试样本x与训练集样本X_i之间的距离，通常使用欧氏距离公式：

$$d(x,X_i)=\sqrt{\sum_{j=1}^{p}(x_j-X_{ij})^2}\quad(i=1,2,\ldots,n) \tag{1}$$

2. 对于距离x最近的前k个训练集样本，统计它们属于不同类别的出现次数。

3. 投票得出x所属的类别，即出现次数最多的类别。

KNN简单易懂、易于实现、对于非线性和复杂数据具有较好的适应性。

2.2.2 代码部分

本部分共分`FR_holistic.m`主程序，及`svm.m`， `knn.m`两个子程序

1. **主程序代码:**

```
close all;
clear all;
clc;

% 载入训练集
load ORL_trainset;
[dim, trainnum] = size(train_data); % dim为样本维数，trainnum为训练集样本数
classnum = length(unique(train_label)); % 类别数
trainnum_eachclass = trainnum / classnum; % 每类目标训练样本数

% 载入测试集
load ORL_testset;
testnum = size(test_data, 2); % 测试集样本数
testnum_eachclass = testnum / classnum; % 每类目标测试样本数

% 载入测试集标签
load ORL_testlabel;

%-------------------------- 数据标准化或归一化 --------------------------%
%本部分为可选项,经过验证,在OLR数据集中使用整体性方法，提升并不明显
% 标准化
%train_data = zscore(train_data);
%test_data = zscore(test_data);

% 归一化
%train_data = normalize(train_data);
%test_data = normalize(test_data);
%-------------------------- 数据标准化或归一化 --------------------------%

%---------------------------- 训练分类器并测试 ---------------------------%
% 训练SVM分类器并测试
svm_predicted_labels = svm(train_data, train_label, test_data);

% 训练KNN分类器并测试
k = 1; % 设定KNN算法中的K值
knn_predicted_labels = knn(train_data, train_label, test_data, k);
%---------------------------- 训练分类器并测试 ---------------------------%
```

```
%----------------------------计算分类精度----------------------------%
svm_accuracy = calculate_accuracy(svm_predicted_labels, label_truth);
knn_accuracy = calculate_accuracy(knn_predicted_labels, label_truth);

svm_accuracy = 100 * svm_accuracy;          % 将分类精度以百分数的形式输出
knn_accuracy = 100 * knn_accuracy;          % 将分类精度以百分数的形式输出

fprintf('使用SVM分类器在测试集上的分类精度为 %.2f%%\n', svm_accuracy);
fprintf('使用KNN分类器在测试集上的分类精度为 %.2f%%\n', knn_accuracy);
%----------------------------计算分类精度----------------------------%
```

2. `svm` 子程序：

```
function predicted_labels = svm(train_data, train_label, test_data)
    svm_model = fitcecoc(train_data', train_label); % 使用fitcecoc训练多类别分类器
    predicted_labels = predict(svm_model, test_data');
end
```

3. `knn` 子程序：

```
function predicted_labels = knn(train_data, train_label, test_data, k)
    knn_model = fitcknn(train_data', train_label, 'NumNeighbors', k);
    predicted_labels = predict(knn_model, test_data');
end
```

2.2.3 实验结果

分类器	准确率
KNN	88.12%
SVM	84.38%

表 3 分类结果

该数据集图像特征较为明显，且干扰因素不多，直接采取整体性方法进行分类取得了不错的效果。

2.3 基于特征提取的实现方式

2.3.1 基本原理

2.3.1.1 PCA（主成分分析）

> 主成分分析（Principal Component Analysis，PCA）是一种常用的降维和特征提取方法，用于将高维数据转换为低维表示。PCA的基本原理是通过线性变换将数据投影到新的特征空间，使得投影后的特征具有最大的方差。[2]

PCA（Principal Component Analysis）可以概括为以下几个步骤：

1. 将数据矩阵X进行中心化处理（即减去每个维度的均值），得到中心化矩阵$\tilde{X}$。
2. 计算协方差矩阵$S = \frac{1}{n}\tilde{X}\tilde{X}^T$。

3. 对协方差矩阵S进行特征值分解，得到特征值和对应的特征向量。将特征向量按照对应的特征值大小从大到小排序，得到特征向量矩阵V。

4. 选取前k个特征向量构成投影矩阵W。将中心化矩阵$\tilde{X}$乘以投影矩阵W，得到降维后的数据矩阵$Y = W^T\tilde{X}$。

   协方差矩阵S的计算公式为：$S = \frac{1}{n}\tilde{X}\tilde{X}^T = \frac{1}{n}(X-\bar{x})(X-\bar{x})^T = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(x_i-\bar{x})^T$

经过PCA降维，可以实现对高维数据的压缩和去除冗余信息，同时保留主要的特征。降维后的数据可以用于可视化、数据压缩、特征提取和分类等任务。

2.3.1.2 LDA（线性判别分析）

> 线性判别分析（Linear Discriminant Analysis，LDA）是一种经典的监督学习方法，主要用于降低维度和提取判别特征，尤其适用于分类问题。LDA的基本原理是将数据投影到一个低维子空间，使得同一类别的样本尽可能接近，不同类别的样本尽可能分开。[2]

基于PCA的方法的一个问题是：投影将训练集中所有图像的方差最大化，这意味着顶级特征向量可能会对识别精度产生负面影响，因为它们可能对应于与识别任务无关的内部变化（如光照、姿势或表情）。LDA可以很好的解决这个问题。

线性判别分析（Linear Discriminant Analysis，LDA）是一种经典的有监督线性降维算法。下面是LDA的数学推导过程：

1. 对于一个d维的样本集合$\{x_1, x_2, \ldots, x_n\}$，其中$x_i \in \mathbb{R}^d$，我们的目标是寻找一个$d' \leq d$的投影方向$w$，将样本映射到一条直线上，使得原始数据在这个方向上的投影尽可能地区分不同类别。

2. 将样本集合按类别分开，用$\{x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}\}$表示第$i$个类别的样本集合，其中$n_i$为第$i$个类别的样本数。

3. 计算每个类别的样本均值向量m_i：$m_i = \frac{1}{n_i}\sum_{x\in X_i} x$ (其中X_i为第i个类别的样本集合)。

4. 计算类内散度矩阵S_w和类间散度矩阵S_b：$S_w = \sum_{i=1}^{K}\sum_{x\in X_i}(x-m_i)(x-m_i)^T \quad S_b = \sum_{i=1}^{K} n_i(m_i-m)(m_i-m)^T$

其中K为类别数，m为所有样本的均值向量，即$m = \frac{1}{n}\sum_{i=1}^{K}\sum_{x\in X_i} x$。

5. 求解广义瑞利商$\frac{w^T S_b w}{w^T S_w w}$的最大特征值$\lambda$及其对应的特征向量$w$。$S_w$为类内散度矩阵，$S_b$为类间散度矩阵。

6. 将样本映射到w方向上，得到降维后的样本$x' = w^T x$。

通过以上步骤，LDA可以得到一组特征向量，这些特征向量具有判别性，能够最大化类间散度并最小化类内散度。投影到这些特征向量上的数据能够更好地区分不同类别，从而提高分类性能。

2.3.2 代码部分

本部分共分为`FR_featurebased.m`主程序及`svm.m`，`knn.m`，`pca.m`，`lda.m`四个子程序，其中`svm.m`，`knn.m`同上，不在展示。

1. **主程序代码:**

```
close all;
clear all;
```

```
clc;

% 载入训练集
load ORL_trainset;
[dim, trainnum] = size(train_data); % dim为样本维数，trainnum为训练集样本数
classnum = length(unique(train_label)); % 类别数
trainnum_eachclass = trainnum / classnum; % 每类目标训练样本数

% 载入测试集
load ORL_testset;
testnum = size(test_data, 2); % 测试集样本数
testnum_eachclass = testnum / classnum; % 每类目标测试样本数

% 载入测试集标签
load ORL_testlabel;

%------------------------ 网格搜索 ------------------------%
% 设置网格搜索的参数范围
pca_dims_range = 1:100; % PCA降维维度的范围
lda_dims_range = 1:39; % LDA降维维度的范围
k_range = 1:5; % KNN中K值的范围

svm_pca_best_accuracy = 0;
svm_pca_best_dims = 0;
svm_lda_best_accuracy = 0;
svm_lda_best_dims = 0;
knn_pca_best_accuracy = 0;
knn_pca_best_dims = 0;
knn_pca_best_k = 0;
knn_lda_best_accuracy = 0;
knn_lda_best_dims = 0;
knn_lda_best_k = 0;

svm_pca_log = [];
svm_lda_log = [];
knn_pca_log = [];
knn_lda_log = [];

% 进行网格搜索
for pca_dims = pca_dims_range
    % 特征提取降维
    [train_pca, test_pca] = pca(train_data, test_data, pca_dims);
    % 训练SVM分类器并测试
    svm_pca_predicted_labels = svm(train_pca, train_label, test_pca);
    % 计算分类精度
    svm_pca_accuracy = calculate_accuracy(svm_pca_predicted_labels, label_truth);

    svm_pca_log = [svm_pca_log; pca_dims svm_pca_accuracy];
```

```
    if svm_pca_accuracy > svm_pca_best_accuracy
        svm_pca_best_accuracy = svm_pca_accuracy;
        svm_pca_best_dims = pca_dims;
    end
    fprintf('SVM分类器中PCA维度为 %d 的准确度为 %.2f%%\n', pca_dims,
svm_pca_accuracy*100);
end
fprintf('SVM分类器中最优的PCA维度为 %d，对应准确度为 %.2f%%\n', svm_pca_best_dims,
svm_pca_best_accuracy*100);
% 保存日志
log_folder = './Grid_search_log';
save(fullfile(log_folder, 'svm_pca_log.mat'), 'svm_pca_log');
for lda_dims = lda_dims_range
    % 特征提取降维
    [train_lda, test_lda] = lda(train_data, train_label, test_data, lda_dims);
    % 训练SVM分类器并测试
    svm_lda_predicted_labels = svm(train_lda, train_label, test_lda);
    % 计算分类精度
    svm_lda_accuracy = calculate_accuracy(svm_lda_predicted_labels, label_truth);

    svm_lda_log = [svm_lda_log; lda_dims svm_lda_accuracy];

    if svm_lda_accuracy > svm_lda_best_accuracy
        svm_lda_best_accuracy = svm_lda_accuracy;
        svm_lda_best_dims = lda_dims;
    end
    fprintf('SVM分类器中LDA维度为 %d 的准确度为 %.2f%%\n', lda_dims,
svm_lda_accuracy*100);
end
fprintf('SVM分类器中最优的LDA维度为 %d，对应准确度为 %.2f%%\n', svm_lda_best_dims,
svm_lda_best_accuracy*100);

for pca_dims = pca_dims_range
    % 特征提取降维
    [train_pca, test_pca] = pca(train_data, test_data, pca_dims);
    for k = k_range
        % 训练KNN分类器并测试
        knn_pca_predicted_labels = knn(train_pca, train_label, test_pca, k);
        % 计算分类精度
        knn_pca_accuracy = calculate_accuracy(knn_pca_predicted_labels,
label_truth);

        knn_pca_log = [knn_pca_log; pca_dims k knn_pca_accuracy];

        if knn_pca_accuracy > knn_pca_best_accuracy
            knn_pca_best_accuracy = knn_pca_accuracy;
            knn_pca_best_dims = pca_dims;
```

```
            knn_pca_best_k = k;
        end
        fprintf('KNN分类器中PCA维度为 %d, K值为 %d 的准确度为 %.2f%%\n', pca_dims, k,
knn_pca_accuracy*100);
    end
end
fprintf('KNN分类器中最优的PCA维度为 %d, K值为 %d, 对应准确度为 %.2f%%\n',
knn_pca_best_dims, knn_pca_best_k, knn_pca_best_accuracy*100);

for lda_dims = lda_dims_range
    % 特征提取降维
    [train_lda, test_lda] = lda(train_data, train_label, test_data, lda_dims);
    for k = k_range
        % 训练KNN分类器并测试
        knn_lda_predicted_labels = knn(train_lda, train_label, test_lda, k);
        % 计算分类精度
        knn_lda_accuracy = calculate_accuracy(knn_lda_predicted_labels,
label_truth);

        knn_lda_log = [knn_lda_log; lda_dims k knn_lda_accuracy];

        if knn_lda_accuracy > knn_lda_best_accuracy
            knn_lda_best_accuracy = knn_lda_accuracy;
            knn_lda_best_dims = lda_dims;
            knn_lda_best_k = k;
        end
        fprintf('KNN分类器中LDA维度为 %d, K值为 %d 的准确度为 %.2f%%\n', lda_dims, k,
knn_lda_accuracy*100);
    end
end
fprintf('KNN分类器中最优的LDA维度为 %d, K值为 %d, 对应准确度为 %.2f%%\n',
knn_lda_best_dims, knn_lda_best_k, knn_lda_best_accuracy*100);
%------------------------ 网格搜索 ------------------------%

% 保存日志
log_folder = './Grid_search_log';
save(fullfile(log_folder, 'svm_pca_log.mat'), 'svm_pca_log');
save(fullfile(log_folder, 'svm_lda_log.mat'), 'svm_lda_log');
save(fullfile(log_folder, 'knn_pca_log.mat'), 'knn_pca_log');
save(fullfile(log_folder, 'knn_lda_log.mat'), 'knn_lda_log');
```

2. `pca.m`子程序

```
function [train_pca, test_pca] = pca(train_data, test_data, num_dims)
    % 计算训练集的均值
    train_mean = mean(train_data, 2);
```

```
    % 中心化训练集和测试集
    train_centered = train_data - train_mean;
    test_centered = test_data - train_mean;

    % 计算训练集的协方差矩阵
    cov_matrix = cov(train_centered');

    % 对协方差矩阵进行特征值分解
    [eig_vectors, eig_values] = eig(cov_matrix);

    % 对特征值进行排序并选择前num_dims个特征向量
    [~, sorted_idx] = sort(diag(eig_values), 'descend');
    selected_eig_vectors = eig_vectors(:, sorted_idx(1:num_dims));

    % 对训练集和测试集进行降维
    train_pca = selected_eig_vectors' * train_centered;
    test_pca = selected_eig_vectors' * test_centered;
end
```

3. **`lda.m`子程序**

```
function [train_lda, test_lda] = lda(train_data, train_label, test_data,
num_dims_lda)
    % 计算每个类别的样本均值
    class_means = zeros(size(train_data, 1), max(train_label));

    for i = 1:max(train_label)
        class_means(:, i) = mean(train_data(:, train_label == i), 2);
    end

    % 计算总体均值
    overall_mean = mean(train_data, 2);

    % 计算类内散布矩阵
    Sw = zeros(size(train_data, 1), size(train_data, 1));

    for i = 1:max(train_label)
        class_data = train_data(:, train_label == i);
        class_centered = class_data - class_means(:, i);
        Sw = Sw + class_centered * class_centered';
    end

    % 计算类间散布矩阵
    Sb = zeros(size(train_data, 1), size(train_data, 1));

    for i = 1:max(train_label)
        class_centered = class_means(:, i) - overall_mean;
```

```
        Sb = Sb + size(train_data(:, train_label == i), 2) * (class_centered *
                class_centered');
    end

    % 添加正则化项
    lambda = 0.001;
    Sw_reg = Sw + lambda * eye(size(Sw));

    % 修正类内散布矩阵的特征值
    epsilon = 1e-6;
    Sw_reg = Sw_reg + epsilon * trace(Sw_reg) * eye(size(Sw_reg));

    % 对 (Sw_reg^-1) * Sb 进行特征值分解
    [eig_vectors, eig_values] = eig(inv(Sw_reg) * Sb);

    % 对特征值进行排序并选择前 num_dims_lda 个特征向量
    [~, sorted_idx] = sort(diag(eig_values), 'descend');
    selected_eig_vectors = eig_vectors(:, sorted_idx(1:num_dims_lda));

    % 对训练集和测试集进行降维
    train_lda = selected_eig_vectors' * train_data;
    test_lda = selected_eig_vectors' * test_data;
end
```

2.3.3 实验结果

使用PCA和LDA技术进行特征提取和降纬后，分别使用SVM分类器和KNN分类器进行分类，通过网格搜索技术，寻找PCA和LDA降低的最优纬度以及KNN算法中K的最优解，完成分类任务。网格搜索结果和分类正确率如下。

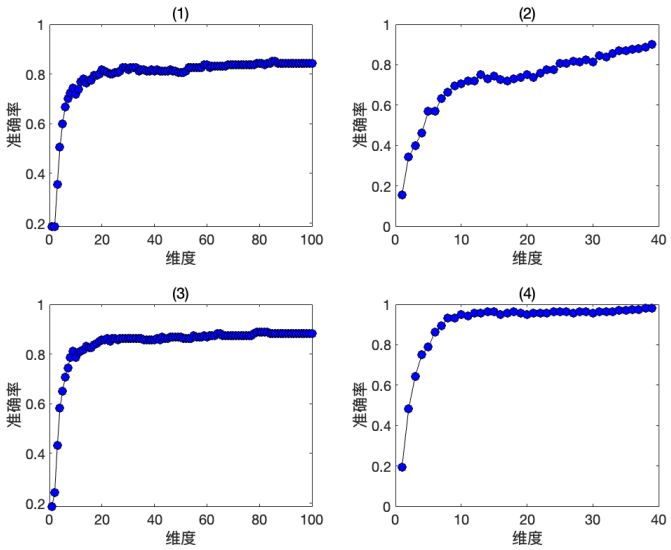

图 4 （1）SVM-PCA （2）SVM-LDA （3）KNN-PCA （4）KNN-LDA

采用最优参数训练分类器的准确率为：

方法	准确率
SVM+PCA	85.00%
SVM+LDA	90.00%
KNN+PCA	88.75%
KNN+LDA	98.12%

表 4 分类结果

可以看出，通过提取特征，结合分类器进行分类，取得了良好的分类效果。

2.4 基于深度学习的实现方式

深度学习是一类使用多层线性及非线性处理单元通过组合底层特征而形成更加抽象的高层特征表示的机器学习算法，基于深度学习的人脸识别方法使用端到端的方式学习提取特征的能力，并使用提取到的特征进行分类。在损失函数的指导下利用一些优化方法如随机梯度下降、自适应学习率算法优化神经网络中的参数。深度学习通过构建深层神经网络模型，自动从图像或视频中学习特征表示，完成人脸识别任务。

本部分参考了西安交通大学余璀璨教授《基于深度学习的人脸识别方法综述》[3]。

2.4.1 基本原理

2.4.1.1 CNN(卷积神经网络)

卷积神经网络（Convolutional Neural Network，CNN）是一种深度学习模型，用于处理具有网格结构的数据，例如图像。它通过在网络中引入卷积层和池化层，能够有效地捕捉图像中的局部特征和空间结构。

下面是卷积神经网络的基本结构：

1. 输入层：接受原始图像作为输入。
2. 卷积层（Convolutional Layer）：卷积层是CNN的核心组件之一。它由多个卷积核组成，每个卷积核负责提取图像中的某种特定特征。卷积操作在输入图像上滑动卷积核，通过逐元素乘法和求和运算，生成特征图（Feature Map）。卷积层的参数包括卷积核的大小、步幅（Stride）和填充（Padding）方式。
3. 激活函数层（Activation Layer）：激活函数引入非线性变换，增加模型的表达能力。常用的激活函数包括ReLU（Rectified Linear Unit）、Sigmoid和Tanh等。
4. 池化层（Pooling Layer）：池化层用于减小特征图的空间尺寸，同时保留重要的特征。常见的池化操作有最大池化（Max Pooling）和平均池化（Average Pooling）。
5. 全连接层（Fully Connected Layer）：全连接层将前面的层的输出连接到一个全连接的神经网络中。全连接层的神经元可以学习到更高级的特征表示，并将其映射到最终的输出类别或标签。
6. 输出层：输出层给出最终的分类结果，可以使用Softmax函数将网络的输出转化为概率分布，表示每个类别的概率。

卷积神经网络通过多次堆叠卷积层、激活函数层、池化层和全连接层，形成一个深层网络结构。深度学习的优势在于它能够自动学习图像中的特征表示，从而在人脸识别等任务中取得较好的性能。同时，卷积神经网络还可以通过反向传播算法进行端到端的训练，无需手工设计特征提取器。这使得卷积神经网络成为当前人脸识别和图像识别领域的主流方法之一。

2.4.2 代码部分

本部分共分为`FR_deeplearning.m`主程序及`myCNN.m`子程序。

1. **主程序代码**

```
close all;
clear all;
clc;

% 载入训练集
load ORL_trainset;
[dim, trainnum] = size(train_data); % dim为样本维数，trainnum为训练集样本数
classnum = length(unique(train_label)); % 类别数
trainnum_eachclass = trainnum / classnum; % 每类目标训练样本数

% 载入测试集
load ORL_testset;
testnum = size(test_data, 2); % 测试集样本数
testnum_eachclass = testnum / classnum; % 每类目标测试样本数

% 载入测试集标签
load ORL_testlabel;

%-------------------------- 数据标准化或归一化 --------------------------%
% 本部分为可选项，根据需要选择是否进行数据标准化或归一化处理
% 标准化
%train_data = zscore(train_data);
%test_data = zscore(test_data);

% 归一化
train_data = normalize(train_data);
test_data = normalize(test_data);
%-------------------------- 数据标准化或归一化 --------------------------%

%---------------------------- 训练卷机神经网络并测试 -------------------------%
% 训练卷积神经网络并测试
cnn_predicted_labels = myCNN(train_data, train_label, test_data, trainnum);
%---------------------------- 训练卷机神经网络并测试 -------------------------%

%----------------------------计算分类精度-----------------------------%
cnn_accuracy = calculate_accuracy(cnn_predicted_labels, label_truth);
cnn_accuracy = 100 * cnn_accuracy; % 将分类精度以百分数的形式输出
fprintf('使用自建卷积神经网络在测试集上的分类精度为 %.2f%%\n', cnn_accuracy);
```

```
%----------------------------计算分类精度----------------------------%
```

2. `myCNN.m`子程序

```
function predicted_labels = myCNN(train_data, train_label, test_data, trainnum)
    % 使用卷积神经网络进行训练和预测

    % 构建卷积神经网络模型
    layers = [
        imageInputLayer([32 32 1]) % 输入层，指定输入图像的尺寸
        convolution2dLayer(5, 32) % 卷积层，使用5x5的卷积核，输出32个特征图
        reluLayer() % ReLU激活函数层
        maxPooling2dLayer(2, 'Stride', 2) % 最大池化层，使用2x2的窗口进行池化
        fullyConnectedLayer(40) % 全连接层，输出40个类别
        softmaxLayer() % Softmax层，进行分类
        classificationLayer() % 分类层
    ];

    % 设置训练参数
    options = trainingOptions('adam', 'MaxEpochs', 600, 'MiniBatchSize', 32,
'Verbose', true, ...
        'Plots', 'training-progress', 'OutputFcn', @myTrainingProgressFcn);

    % 将训练数据转换为图像数据格式
    XTrain = reshape(train_data, [32 32 1 trainnum]);

    % 将训练标签转换为分类标签数据格式
    YTrain = categorical(train_label);

    % 创建用于保存训练过程数据的结构体
    myCNN_log = struct('Epoch', [], 'Accuracy', []);

    % 自定义训练过程回调函数
    function stop = myTrainingProgressFcn(info)
        % 在每个轮次结束时记录准确率数据
        myCNN_log.Epoch = [myCNN_log.Epoch, info.Epoch];
        myCNN_log.Accuracy = [myCNN_log.Accuracy, info.TrainingAccuracy];
        stop = false; % 继续训练
    end

    % 训练卷积神经网络模型
    net = trainNetwork(XTrain, YTrain, layers, options);

    % 将测试数据转换为图像数据格式
    XTest = reshape(test_data, [32 32 1 size(test_data, 2)]);

    % 使用训练好的模型进行预测
```

```
    YPred = classify(net, XTest);

    % 将预测结果转换为数值标签格式
    predicted_labels = double(YPred);

    % 绘制正确率随轮次的图像
    %figure;
    %plot(myCNN_log.Epoch, myCNN_log.Accuracy);
    %xlabel('Epoch');
    %ylabel('Accuracy');
    %title('Accuracy vs. Epoch');
end
```

2.4.3 实验结果

通过构建一个简单的卷积神经网络，拟合人脸特征，完成分类。

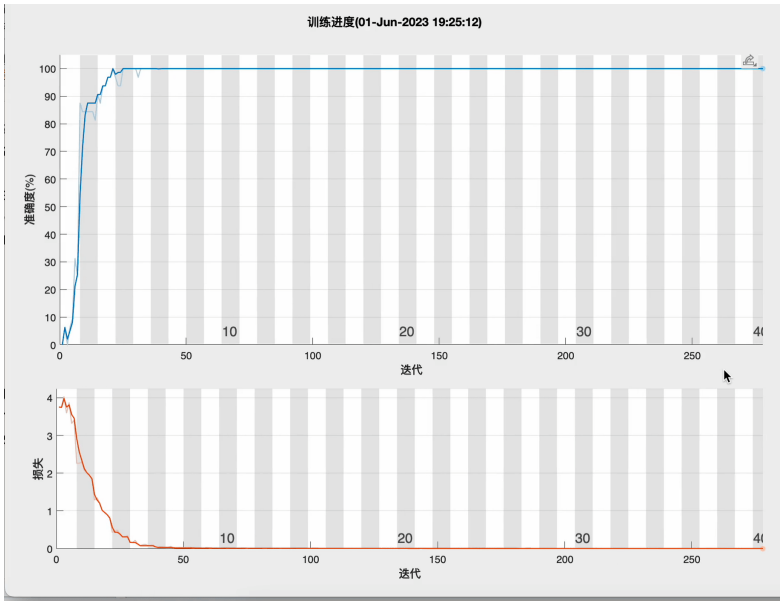

图 5 训练轮次图

经过250轮迭代，逐渐收敛，分类结果如下：

方法	准确率
卷积神经网络	95.62%

表 5 分类结果

本次实验仅训练了一个简单的神经网络，经过迭代后分类正确率达到了95.62%，可见深度学习处理人脸识别的强大之处。

3 后记

实验结果汇总:

方法	准确率
SVM	84.38%
KNN	88.12%
SVM+PCA	85.00%
SVM+LDA	90.00%
KNN+PCA	88.75%
KNN+LDA	98.12%
CNN	95.62%

表 6 分类结果

本次实验分别使用了整体性方法，特征提取方法以及深度学习方法三种方法对人脸进行分类；由上表数据，特征提取后使用分类器进行分类效果明显提升，特别是使用KNN+LDA算法时正确率达到了98.12%，很好的完成了任务；同时，在本身网络结构比较简单的情况下，卷积神经网络算法也取得了95.62%的效果，可以预见，通过调整网络结构和参数，卷机神经网络在人脸识别任务中潜力很大。

开源地址：https://github.com/anglee2002/PatternRecognition/

参考文献:

[1] Daniel Sáez Trigueros, Meng L, Hartnett M. Face Recognition: From Traditional to Deep Learning Methods

[2] wikipedia. 基于几何的人脸识别[EB/OL]. [2018-10-04]

[3] 余璀璨, 李慧斌. 基于深度学习的人脸识别方法综述[J]. 工程数学学报 , Aug. 2021, 第38卷(第4期)