实　　验　　报　　告

学　　号	21050005076	姓　　名	田晶怡	专业班级	2021 级智能科学与技术
课程名称	机器学习			学期	2024 年 春季学期
任课教师	仲国强	完成日期	2024/6/4	上机课时间	2024/5/27
实 验 名 称	基于机器学习算法的人脸识别系统				

一、 实验目的

1. 实现基于支持向量机的人脸识别方法、基于卷积神经网络的人脸识别方法等（至少实现两种方法）。

2. 采用多个性能指标评价（不限于准确率 accuracy、F1 值等）进行方法对比。

二、 实验原理：

基于机器学习的人脸识别系统

基于机器学习的人脸识别系统是一种利用计算机视觉和模式识别技术来自动识别和验证人脸的系统。它的原理涉及三个主要步骤：

1.人脸检测：该步骤旨在从图像或视频中检测出人脸的位置。常用的方法包括基于 Haar 特征的级联分类器、基于深度学习的卷积神经网络(CNN)等。这些方法通过训练一个分类器来判断某个区域是否为人脸。

2.特征提取：在此步骤中，系统从检测到的人脸中提取出具有区分性的特征向量。传统的方法使用的是基于人工设计的特征，如局部二进制模式(Local Binary Patterns，LBP)、主成分分析(Principal Component Analysis，PCA)等。而现代的方法则借助深度学习模型(如 FaceNet、ArcFace)来学习高层次的表征。

3.人脸识别：在这一步骤中，使用提取到的人脸特征，训练分类器。随后根据输入的人脸数据进行识别并输出识别结果。

基本实现流程

1.数据收集:首先需要收集大量的人脸图像数据集。这些数据集应包含具有不同人物的正面人脸图像，并且要求图像质量和角度差异较大，以覆盖各种情况。

2.数据预处理:对采集到的人脸图像进行预处理，包括图像去躁、裁剪和调整大小等操作。此外，还可以使用直方图均衡化或归一化来增强图像的对比度和亮度。

3.人脸检测:利用人脸检测算法，如基于 Haar 特征的级联分类器、基于深度学习的卷积神经网络(CNN)、YOLO 等，来检测输入图像中的人脸位置。

4.特征提取:使用特定的特征提取算法从每个检测到的人脸中提取出具有区分性的特征向量。常用的方法包括局部二进制模式(Local Binary Patterns，LBP)、主成分分析(PrincipalComponent Analysis，PCA)和深度学习模型(如 FaceNet、ArcFace)。特征提取的目标是将人脸图像转换为高维特征向量，保留人脸的关键信息。

5.训练分类器:使用提取到的人脸特征向量和对应的标签(人物身份)来训练一个分类器模型，如支持向量机(Support Vector Machines，SVM)、k 近邻(k-Nearest Neighbors，k-NN)等。分类器模型将学习如何根据输入的特征向量进行人脸识别。

6.测试:在测试阶段，将待识别的人脸图像通过相同的预处理、人脸检测和特征提取步骤获取特征向量。然后，利用训练好的分类器模型将提取到的特征向量进行分类（识别）。

三、 实验内容：

1. 数据集介绍

本次实验我主要使用两个数据集，一个是 sklearn 库自带的一个数据集 fetch_lfw_people，另一个是我自己收集的数据集 MyImage。接下来简要介绍。

fetch_lfw_people（）数据集：

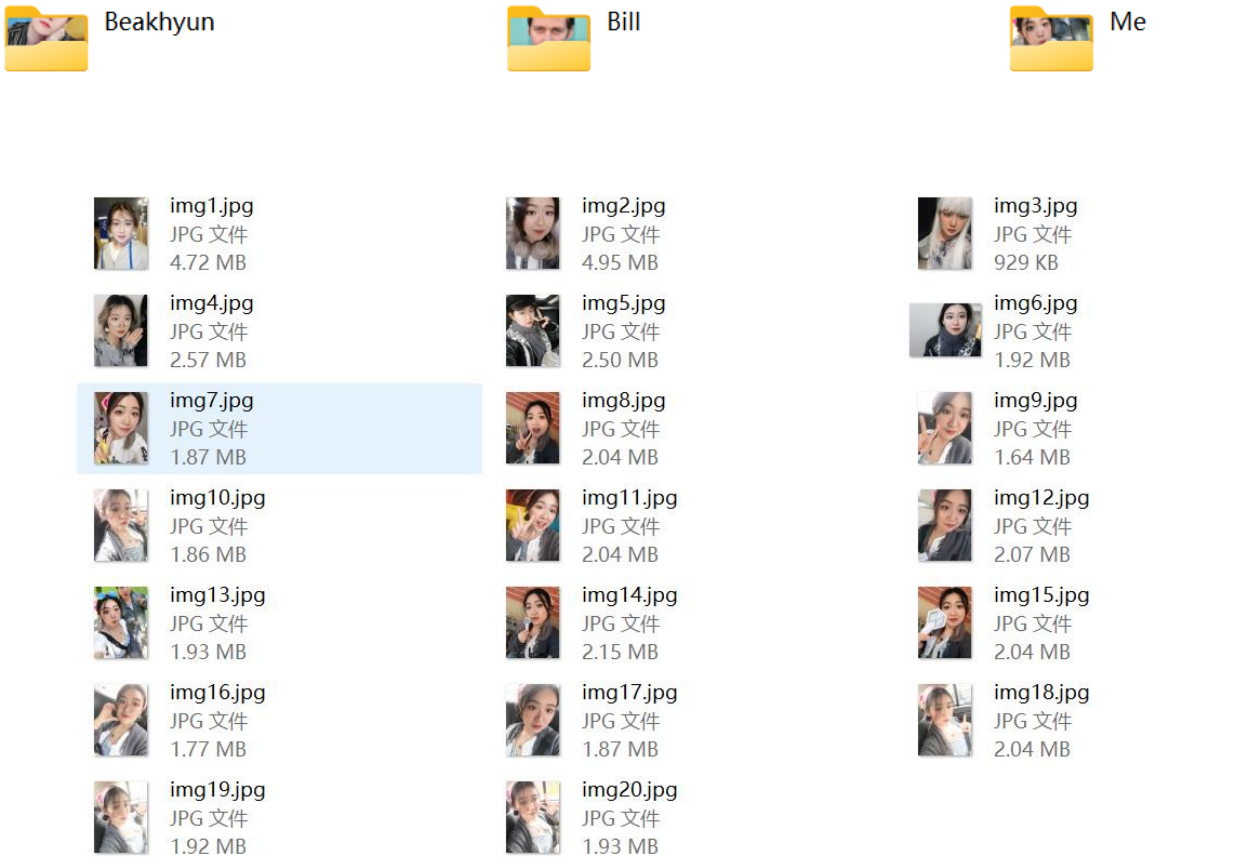
```
lfw_people = fetch_lfw_people()
print(dir(lfw_people))
# ['DESCR', 'data', 'images', 'target', 'target_names']
print(lfw_people.DESCR)
```

```
**Data Set Characteristics:**

=================   =======================
Classes                                5749
Samples total                         13233
Dimensionality                         5828
Features            real, between 0 and 255
=================   =======================
```

输出信息很多，这里就不一一展示了。可以看到，该数据集共 5749 类，即一共有 5749 人，样本总数为 13233，每张照片维度为 5828，特征范围是 0-255。该数据集有两种使用模式，第一种用于人脸识别任务，多分类任务，导入时需要在数据集中表明参数（在代码中有体现），即进行数据过滤；第二种模式通常用于面部验证任务，每个样本是一对照片，判断属不属于一个人。本实验采用第一种使用方法。

MyImage()数据集：



这里是我自己收集的三个人（包括我自己）的面部图像，分成了 train 和 test 两个文件夹，训

练集是 3*20，测试集是 3*4，比较小的数据集（不主要用于模型训练，仅用于模型的使用展示，即好玩）。

2. 基于 SVM 的人脸识别系统

主要代码：

```
faces = fetch_lfw_people(min_faces_per_person=60) # 只识别最少有60张照片的人的人脸
print(faces.target_names)  #输出符合条件的人的姓名
print(faces.images.shape)  #输出照片尺寸
fig, ax = plt.subplots(3,5)  #以3行5列的形式显示照片
for i,axi in enumerate(ax.flat):
  axi.imshow(faces.images[i],cmap = 'bone')
  axi.set(xticks=[],yticks=[],xlabel=faces.target_names[faces.target[i]])
plt.show()

pca = PCA(n_components=150,whiten=True, random_state=42) #PCA降维，维数降到150
svc = SVC(kernel='rbf',class_weight='balanced')  # 使用rbf核函数
model = make_pipeline(pca, svc)#流水线处理，先降维，再SVC分类

Xtrain, Xtest,ytrain,ytest = train_test_split(faces.data, faces.target, random_state=40)
param_grid = {'svc__C': [1, 5, 10], 'svc__gamma': [0.0001, 0.0005, 0.001]}
grid = GridSearchCV(model, param_grid)
print(Xtrain.shape, ytrain.shape) #特征集1011行，2914列，标签集1011行
grid.fit(Xtrain, ytrain)  #建立模型
print(grid.best_params_)  #输出模型的最优参数组合

model = grid.best_estimator_  #获得最好的模型
yfit = model.predict(Xtest)  #用当前最好的模型做预测
fig, ax = plt.subplots(4, 6)
for i, axi in enumerate(ax.flat):
    axi.imshow(Xtest[i].reshape(62, 47), cmap='bone') #每个图像大小是62*47
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel(faces.target_names[yfit[i]].split()[-1],
                   color='black' if yfit[i] == ytest[i] else 'red')
plt.suptitle('Predicted Names; Incorrect Labels in Red', size=14)
plt.show()
```

首先获取至少有 60 张人脸图像的类作为训练集（训练集太少的话模型性能一般）。

```
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
 'Gerhard Schroeder' 'Hugo Chavez' 'Junichiro Koizumi' 'Tony Blair']
(1348, 62, 47)
```

得出符合条件的有 8 个类别，共 1348 张图。

然后调用 sklaern 库，先进行 PCA 降维进行特征提取，将数据集维度 5828 降到 150。然后调用 SVM 模型创建分类器，class_weight='balanced'表示为每个类赋予一个权重，使得每个类都有相同数量的样本。创建模型 Model（流水线对象），先应用 PCA 进行数据预处理，然后将处理后的数据传递给 SVC 进行分类。优化器采用的是 GridSearchCV 对象，用于在参数网格上搜索最佳参数组合。参数网格 param_grid，用于 GridSearchCV，'svc__C'和'svc__gamma'是 SVC 分类器的两个参数，C 是正则化参数，gamma 是核函数的参数。使用 GridSearchCV 在训练集上搜索最佳参数组合，得出最优的模型 model。然后对测试集进行预测即可。这里我多加了一个“标红判断”，即识别错的人名标红，然后和图片一起展示出来，具有非常好的可视化效果。

3. 基于卷积神经网络的人脸识别系统

```
# 采样，每类80张
a = zip(images, target_name)
image = []
target = []
a0, a1, a2, a3, a4, a5, a6, a7, a8, a9 = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
for x, y in a:
    if y == 0:
        if a0 <= 80:
            image.append(x)
            target.append(y)
            a0 += 1
    elif y == 1:
        if a1 <= 80:
            image.append(x)
            target.append(y)
            a1 += 1
    elif y == 2:
        if a2 <= 80:
            image.append(x)
            target.append(y)
            a2 += 1
    elif y == 3:
        if a3 <= 80:
            image.append(x)
            target.append(y)
            a3 += 1
    elif y == 4:
        if a4 <= 80:
            image.append(x)
            target.append(y)
            a4 += 1
    elif y == 5:
        if a5 <= 80:
            image.append(x)
            target.append(y)
            a5 += 1
    elif y == 6:
        if a6 <= 80:
            image.append(x)
            target.append(y)
```

这段代码是用来对从 LFW 数据集中加载的人脸图像进行采样，确保每个类别的样本数量都是 80 张。LFW 数据集包含了大量的人脸图像，但并非每个类别都有足够多的样本。为了平衡每个类别的样本数量，代码使用了一个循环来检查每个样本的类别，并将其添加到相应的类别列表中，直到每个类别的样本数量达到 80 张为止。

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.Conv1 = nn.Sequential(
            nn.Conv2d(  # ->(1, 125, 95)
                in_channels=1,
                out_channels=16,
                kernel_size=(5, 5),
                stride=1,
                padding=2,  # if stride = 1,padding_size = (kernel_size-1)/2
            ),
            nn.BatchNorm2d(num_features=16),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
        )
        self.Conv2 = nn.Sequential(
            nn.Conv2d(16, 32, (5, 5), 1, 2),
            nn.BatchNorm2d(num_features=32),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
        )
        self.Conv3 = nn.Sequential(
            nn.Conv2d(32, 64, (5, 5), 1, 2),  # ->(16, 14, 14)
            nn.BatchNorm2d(num_features=64),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
        )
        self.Conv4 = nn.Sequential(
            nn.Conv2d(64, 128, (5, 5), 1, 2),  # ->(16, 14, 14)
            nn.BatchNorm2d(num_features=128),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
        )
        self.fc = nn.Sequential(
            nn.Linear(128*7*5, 2048),
            nn.ReLU(inplace=True),
            nn.Linear(2048, 1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 10),
        )
    def forward(self, x):
        x = self.Conv1(x)
        x = self.Conv2(x)
        x = self.Conv3(x)
        x = self.Conv4(x)
        x = x.view(x.size(0), -1)
        output = self.fc(x)
        return output


cnn = CNN()
optimizer = torch.optim.Adam(cnn.parameters(), lr=0.001)
loss_function = nn.CrossEntropyLoss()  # 回归问题用MSEloss，分类用CrossEntropyLoss
```

以上的代码是神经网络的结构，我一共定义了 4 个卷积层，和一个全连接层。卷积层中包括一个 2d 卷积层、批量归一化层、激活层和池化层。因为输入的图像是处理过的灰度图，所以第一层卷积层的输入通道数为 1，其他的参数用的都是卷积网络实现的常见量。对于全连接层 fc，其输入特征数量是 Conv4 输出展平后的特征数量，而输出特征为 10，是因为要用来训练网络的数据集

为 10 类。然后通过定义的向前传播函数实现以上各层即可，x = x.view(x.size(0), -1) 将 4D 的卷积层输出转换为 2D 的向量，以便后续的全连接层可以处理。这里的优化器选择 Adam 优化器，用于更新网络参数，损失函数选择交叉熵损失函数 CrossEntropyLoss,用于分类任务。

```
# 训练模型
for epoch in range(EPOCH):
    for step, (x, y) in enumerate(loader):
        b_x = Variable(x)
        b_y = Variable(y)
        if torch.cuda.is_available():
            b_x = b_x.cuda()
            b_y = b_y.cuda()
            cnn = cnn.cuda()
            X_test = X_test.cuda()
            y_test = y_test.cuda()
        output = cnn(b_x)
        loss = loss_function(output, b_y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    # 每个epoch结束时计算准确率
    test_output = cnn(X_test)
    pred_y = torch.max(test_output, 1)[1].data.squeeze()
    accuracy = (pred_y == y_test).sum().float() / y_test.size(0)
    print('Epoch:', epoch, '|train_loss:%.4f' % loss.item(), '|accuracy:', accuracy)
# 保存模型参数
model_path = 'E:/桌面/机器学习/期末实验/My_cnn_model.pth'
torch.save(cnn.state_dict(), model_path)
print(f'Model parameters saved to {model_path}')
```

然后是模型的训练。将当前批次的输入图像 x 和标签 y 转换为 Variable 对象，Variable 对象可以自动求导。接着使用刚刚定义好的模型 cnn 对输入图像进行向前传播，得到网络预测的输出 output，对其进行损失计算。optimizer.zero_grad()清空优化器中的梯度，为下一次反向传播做准备。loss.backward()对损失进行反向传播，计算网络参数的梯度，optimizer.step()使用梯度更新网络参数。每个迭代结束后输出一下训练损失和准确率。

```python
# 已经保存了训练好的模型参数
model_path = 'E:/桌面/机器学习/期末实验/My_cnn_model.pth'

# 加载模型
model = CNN()
model.load_state_dict(torch.load(model_path))

# 定义一个简单的图像预处理步骤
transform = transforms.Compose([
    transforms.Resize((125, 95)),  # 调整图像大小 注意一定要和cnn第一层的输入相同!
    transforms.Lambda(lambda x: x.convert('L')),  # 转换为灰度图
    transforms.ToTensor(),           # 转换为 PyTorch 张量
    transforms.Normalize([0.5],[0.5])
])

# 加载测试数据
image_folder = 'E:/桌面/机器学习/期末实验/CnnImage'
dataset = datasets.ImageFolder(root=image_folder, transform=transform)
loader = DataLoader(dataset, batch_size=1, shuffle=False)

# 初始化准确率和 F1 值
correct = 0
total = 0
predictions = []
targets = []

# 初始化一个列表来存储预测错误的图像
wrong_predictions = []
```

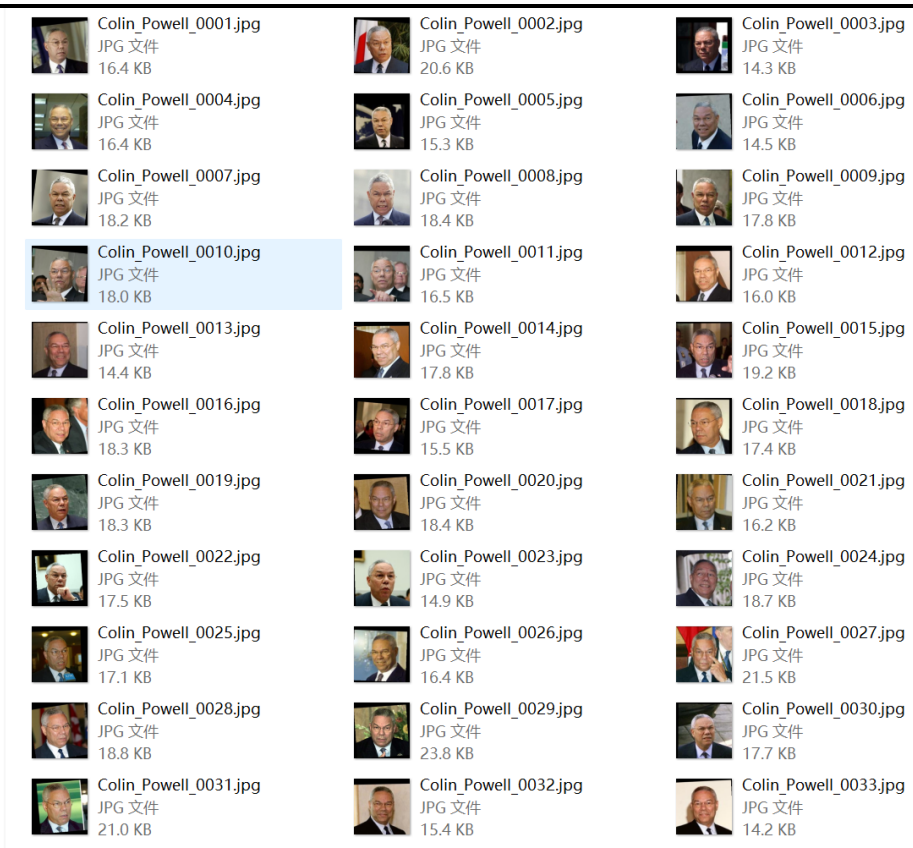
```python
# 运行模型
model.eval()  # 设置模型为评估模式
with torch.no_grad():
    for images, labels in loader:
        if torch.cuda.is_available():
            images = images.cuda()
        #print(images.shape)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        predictions.extend(predicted.cpu().numpy())
        targets.extend(labels.cpu().numpy())

        # 检查预测是否正确
        if predicted != labels:
            wrong_predictions.append((images, predicted, labels))

# 计算准确率和 F1 值
accuracy = accuracy_score(targets, predictions)
f1 = f1_score(targets, predictions, average='macro')

# 打印结果
print(f'Accuracy: {accuracy:.2f}')
print(f'F1 Score: {f1:.2f}')
```

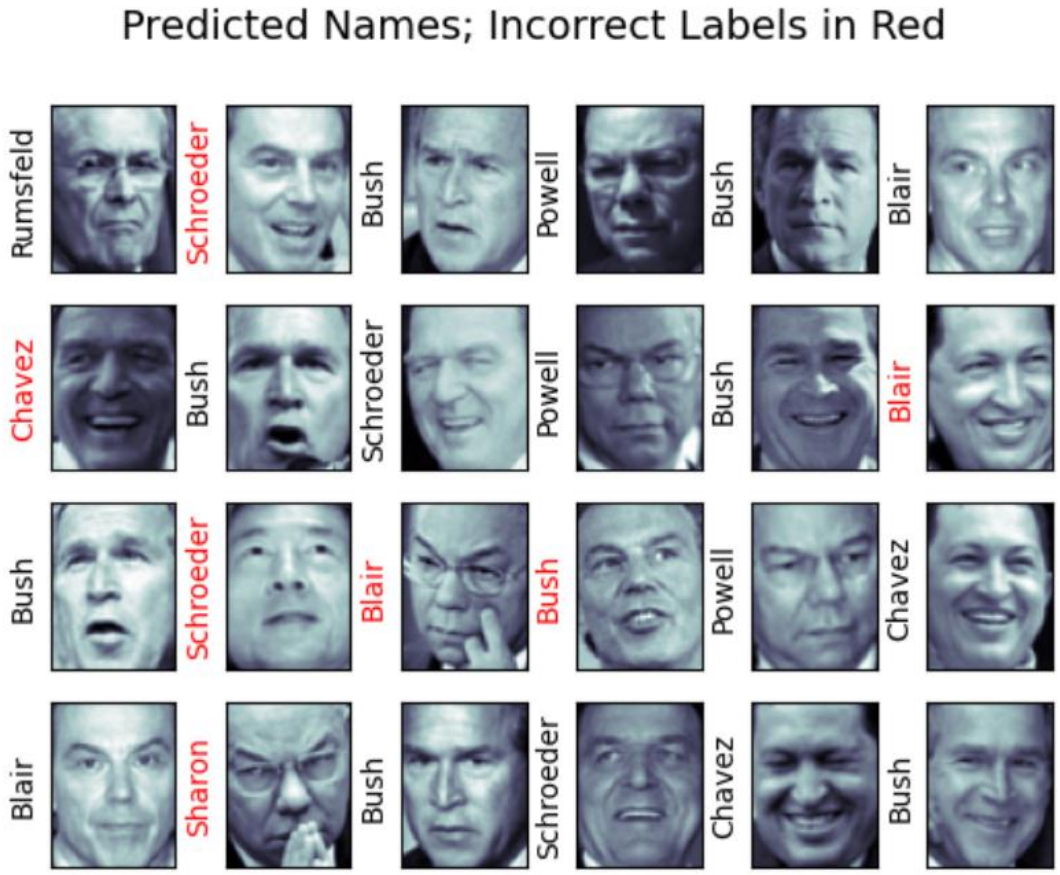
以上代码实现的是模型的一个应用，将上一步训练保存好的模型加载进来，输入的测试图像为单个人物的数据集（训练时作为训练集一部分的人物 Colin Pwell,该人物共有 200 多张图像，训练集取了 80 张，再取没参与过训练的 80 张），看看模型的效果如何。



四、 实验结果及问题：

实验结果：

1. 基于 SVM 的人脸识别系统：



	precision	recall	f1-score	support
Ariel Sharon	0.60	0.56	0.58	16
Colin Powell	0.80	0.83	0.82	54
Donald Rumsfeld	0.78	0.85	0.82	34
George W Bush	0.94	0.89	0.91	136
Gerhard Schroeder	0.68	0.85	0.75	27
Hugo Chavez	0.81	0.72	0.76	18
Junichiro Koizumi	0.87	0.87	0.87	15
Tony Blair	0.83	0.78	0.81	37
accuracy			0.84	337
macro avg	0.79	0.80	0.79	337
weighted avg	0.84	0.84	0.84	337

fetch_lfw_people 数据集结果。可以看到 accuracy 为 84%，每个人物的 precision 和 F1 的值也不错，模型效果较好。

['Beakhyun' 'Bill' 'Me']
(60, 50, 50)

(60, 2500) (60,)
(12, 2500)

	precision	recall	f1-score	support
Beakhyun	1.00	0.50	0.67	4
Bill	1.00	0.25	0.40	4
Me	0.44	1.00	0.62	4
accuracy			0.58	12
macro avg	0.81	0.58	0.56	12
weighted avg	0.81	0.58	0.56	12

MyImage 数据集结果。可以看到，相比起上一个数据集，这次的识别效果就不是很好，accuracy 只有 58%，很有可能是因为数据集太少了的缘故，可见训练数据集的选择对于模型的构建至关重要。

2. 基于卷积神经网络的人脸识别系统：

对多类人物的识别测试：

```
Epoch: 12 |train_loss:0.0295 |accuracy: tensor(0.8739)
Epoch: 13 |train_loss:0.0162 |accuracy: tensor(0.7568)
Epoch: 14 |train_loss:0.0009 |accuracy: tensor(0.8559)
Epoch: 15 |train_loss:0.0021 |accuracy: tensor(0.7117)
Epoch: 16 |train_loss:0.0047 |accuracy: tensor(0.8288)
Epoch: 17 |train_loss:0.0000 |accuracy: tensor(0.8468)
Epoch: 18 |train_loss:0.0001 |accuracy: tensor(0.8559)
Epoch: 19 |train_loss:0.0000 |accuracy: tensor(0.8559)
Epoch: 20 |train_loss:0.0000 |accuracy: tensor(0.8649)
Epoch: 21 |train_loss:0.0023 |accuracy: tensor(0.8559)
Epoch: 22 |train_loss:0.0001 |accuracy: tensor(0.8468)
Epoch: 23 |train_loss:0.0004 |accuracy: tensor(0.8468)
Epoch: 24 |train_loss:0.0015 |accuracy: tensor(0.8468)
Epoch: 25 |train_loss:0.0000 |accuracy: tensor(0.8559)
Epoch: 26 |train_loss:0.0000 |accuracy: tensor(0.8559)
Epoch: 27 |train_loss:0.0000 |accuracy: tensor(0.8468)
Epoch: 28 |train_loss:0.0000 |accuracy: tensor(0.8468)
Epoch: 29 |train_loss:0.0003 |accuracy: tensor(0.8559)
Model parameters saved to E:/桌面/机器学习/期末实验/My_cnn_model.pth
```

迭代 30 次的结果，可以观察到在第 20 次迭代左右，模型的 loss 已经很低了，再继续迭代会过拟合。后面的迭代模型的 accuracy 趋于稳定在 84%-85%，可以看见，这个模型的效果也是不错的。

对单一人物的识别测试：

```
# 打印结果
print(f'Accuracy: {accuracy:.2f}')
print(f'F1 Score: {f1:.2f}')

# 展示预测错误的图像
# for i, (image, predicted, actual) in enumerate(wrong_predictions):
#     # 显示图像和预测结果
#     plt.figure(figsize=(6, 6))
#     #print(image.shape)
#     image = torch.reshape(image, (1, 125, 95))
#     plt.imshow(image.permute(1, 2, 0).detach().cpu().numpy() * 255)
#     plt.text(10, 10, str(predicted.item()), color='red')
#     plt.show()
# target_names = np.unique(targets)
```

```
Accuracy: 0.81
F1 Score: 0.83
```

可以看到 accuracy 和 F1 的值也是挺不错的，由以上可看出此模型的构建较为成功。

3. 问题：

问题 1：寻找最优模型失败

```
ValueError:
All the 45 fits failed.
It is very likely that your model is misconfigured.
You can try to debug the error by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
45 fits failed with the following error:
Traceback (most recent call last):
  File "D:\Anaconda\envs\pytorch\Lib\site-packages\sklearn\model_selection\_validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "D:\Anaconda\envs\pytorch\Lib\site-packages\sklearn\base.py", line 1474, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\Anaconda\envs\pytorch\Lib\site-packages\sklearn\pipeline.py", line 471, in fit
    Xt = self._fit(X, y, routed_params)
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\Anaconda\envs\pytorch\Lib\site-packages\sklearn\pipeline.py", line 408, in _fit
    X, fitted_transformer = fit_transform_one_cached(
```

解决方法：使用 GridSearchCV 工具来搜索最优的 SVM 参数组合失败，因为我自己提供的 Myimage 数据集数量太少。直接用先降维，再 SVC 分类的简单模型处理简单数据集即可。

问题 2：在对单个人物进行识别时，总是显示图像维度和模型不匹配

```
File D:\Anaconda\envs\pytorch\Lib\site-packages\torchvision\transforms\transforms.py:277, in Normalize.fo
    269 def forward(self, tensor: Tensor) -> Tensor:
    270     """
    271     Args:
    272         tensor (Tensor): Tensor image to be normalized.
   (...)
    275         Tensor: Normalized Tensor image.
    276     """
--> 277     return F.normalize(tensor, self.mean, self.std, self.inplace)

File D:\Anaconda\envs\pytorch\Lib\site-packages\torchvision\transforms\functional.py:349, in normalize(te
    346 if not isinstance(tensor, torch.Tensor):
    347     raise TypeError(f"img should be Tensor Image. Got {type(tensor)}")
--> 349 return F_t.normalize(tensor, mean=mean, std=std, inplace=inplace)

File D:\Anaconda\envs\pytorch\Lib\site-packages\torchvision\transforms\_functional_tensor.py:926, in nor
lace)
    924 if std.ndim == 1:
    925     std = std.view(-1, 1, 1)
--> 926 return tensor.sub_(mean).div_(std)

RuntimeError: output with shape [1, 5, 5] doesn't match the broadcast shape [3, 5, 5]
```

解决方法：定义 transforms.Compoes 对输入图像进行预处理，注意调整图像大小和 cnn 第一层卷积层的输入尺寸相同。

```python
# 定义一个简单的图像预处理步骤
transform = transforms.Compose([
    transforms.Resize((125, 95)),   # 调整图像大小 注意一定要和cnn
    transforms.Lambda(lambda x: x.convert('L')),   # 转换为灰度图
    transforms.ToTensor(),            # 转换为 PyTorch 张量
    transforms.Normalize([0.5],[0.5])
])
```

问题 3：进入卷积图片尺寸太小：

```
    166                         return_indices=self.return_indices)

File D:\Anaconda\envs\pytorch\Lib\site-packages\torch\_jit_internal.py:499, in boolean_dispatch.<locals>.fn
    497     return if_true(*args, **kwargs)
    498 else:
--> 499     return if_false(*args, **kwargs)

File D:\Anaconda\envs\pytorch\Lib\site-packages\torch\nn\functional.py:796, in _max_pool2d(input, kernel_s
ion, ceil_mode, return_indices)
    794 if stride is None:
    795     stride = torch.jit.annotate(List[int], [])
--> 796 return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)

RuntimeError: Given input size: (64x1x1). Calculated output size: (64x0x0). Output size is too small
```

解决方法：一开始我尝试用 cnn 模型测试 MyImage 数据集，于是出现上以上报错，根据报错内容来看，是卷积到最后图像卷没了，还是数据集过少的问题，替换为 Colin Pwell 数据集后报错消失。

五、 心得：

通过本次实验，我深入了解了基于 SVM 和卷积神经网络的人脸识别系统的设计和实现。实验中，我首先使用 SVM 算法进行特征提取和分类，通过网格搜索找到最佳参数组合实现模型，然后使用交叉熵损失函数和 Adam 优化器训练卷积神经网络模型。在实验过程中，我注意到了数据预处理的重要性，如归一化和图像增强，以及模型训练过程中的超参数调整。实验结果显示，卷积神经网络模型的准确率略高于 SVM 算法，表明 CNN 在图像识别任务中的优越性。此外，实验中我还学会了使用 PyTorch 框架和使用 sklearn 库进行模型的构建和训练，以及如何保存和加载模型参数。总的来说，本次实验使我更加熟悉了深度学习和图像识别领域的相关技术，也为我今后在这个领域的进一步研究奠定了基础。