611021208 邱創業

一、 如何完成作業的報告

1. 依據資料庫的影像格式，設計一個讀取 pgm 影像檔的函式。
   - 利用 Pillow.Image 讀入.pgm 影像檔，並 show 出相片。

2. 每張人臉影像均為 92x112=10304 的灰階影像，讀取後請將其轉為 10304x1 的向量，即成為一個樣本。
   - 讀入影像檔後，將其轉為 numpy array 並 flatten。

```
def img_to_vector(filename):
    img  = Image.open(filename)
    img_vector = np.array(img).flatten() # 轉為10304x1的向量
    # print(img_vector.size)

    return img_vector
```

3. 資料庫共含有 400 張影像（40 人，每人 10 張），訓練時請只用 200 張（每人取 5 張）。
   - 利用 os.listdir 找出資料夾內所有檔案，並製作 dataset 以及 label。

```
def read_dataset(path):
    file = os.listdir(path) # 獲取path中所有檔案名稱(不含子資料夾)
    dataset = []
    label = []
    for i in file:
        filename = os.listdir(path+i) # 獲取子資料夾內所有檔案名稱
        for j in filename:
            img = img_to_vector(path+i+'/'+j).astype(np.int64) # 呼叫函式, 讀取路徑檔案
            dataset.append(img)
            label.append(i)
    # print(dataset)
    # print(label)

    return np.array(dataset), np.array(label)
```

   - 使用 train_test_split，將 200 張影像檔用於訓練，200 張影像檔用於測試。

```
def split_data(dataset, label):
    X_train, X_test, y_train, y_test = train_test_split(dataset, label, test_size=0.5, random_state=42) # 將資料集切一半

    return X_train, X_test, y_train, y_test
```

4. 利用 PCA 計算此 200 張影像的轉換矩陣，設法將維度從 10304 降至 10, 20, 30, 40, 50 維

- 使用 sklearn. Decomposition.PCA 將影像檔降維

```
def dimensionality_reduction_PCA(n_components, X_train):
    print("Extracting the top %d eigenfaces from %d faces"% (n_components, X_train.shape[0]))

    # run randomized SVD by the method of Halko et al.
    pca = PCA(n_components=n_components, svd_solver='randomized',whiten=True).fit(X_train)

    return pca

def train_text_transform_Model(model, X_train, X_test):
    print("Start Transform")
    X_train_model = model.transform(X_train)
    X_test_model = model.transform(X_test)

    return X_train_model, X_test_model
```

5. 以這些較低維度的樣本訓練出你所學過的任何分類器來進行辨識。

- 使用 SVC 訓練來進行辨識

```
def classification_svc(X_train_model, y_train):
    print("Fitting")
    param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5], 'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }
    clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)
    clf = clf.fit(X_train_model, y_train)

    return clf
```

6. 請比較不同維度的辨識率，並統計出混淆矩陣(Google 一下這是什麼？)

- 使用 matplotlib 來畫出混淆矩陣

```
def plot_confusion_matrix(y_true, y_pred, matrix_title):
    """confusion matrix computation and display"""
    plt.figure(figsize=(9, 9), dpi=100)

    # use sklearn confusion matrix
    cm_array = confusion_matrix(y_true, y_pred)
    plt.imshow(cm_array[:-1, :-1], interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(matrix_title, fontsize=16)

    cbar = plt.colorbar(fraction=0.046, pad=0.04)
    cbar.set_label('Number of images', rotation=270, labelpad=30, fontsize=12)

    true_labels = np.unique(y_true)
    pred_labels = np.unique(y_pred)
    xtick_marks = np.arange(len(true_labels))
    ytick_marks = np.arange(len(pred_labels))

    plt.xticks(xtick_marks, true_labels, rotation=90)
    plt.yticks(ytick_marks, pred_labels)
    plt.tight_layout()
    plt.ylabel('True label', fontsize=14)
    plt.xlabel('Predicted label', fontsize=14)
    plt.tight_layout()

    plt.show()
```

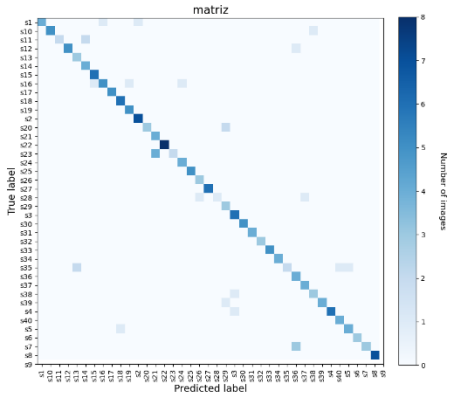
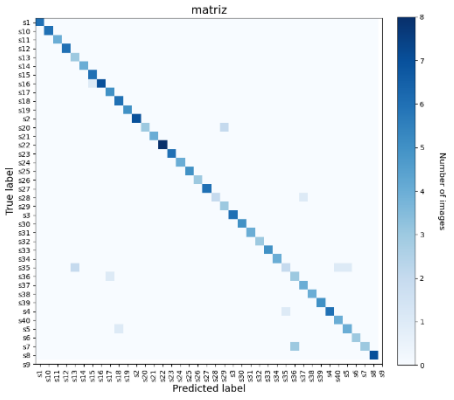
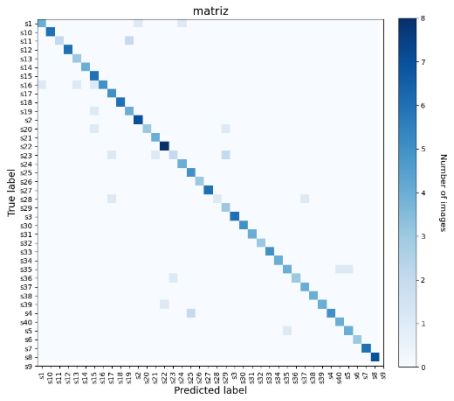
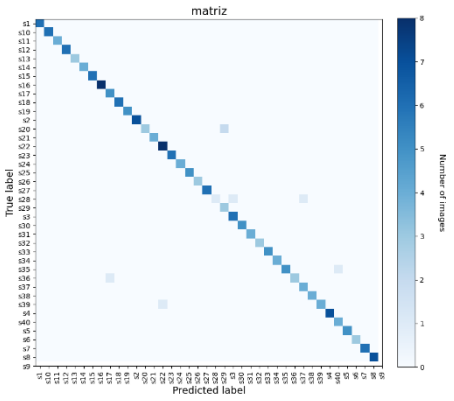
7. 請以降維後的樣本，繼續利用 FLD(LDA)找出另一轉換矩陣，利用此矩陣轉換降維後的樣本（毋需降維只須轉換）為有較佳的類別分離度之新樣本。

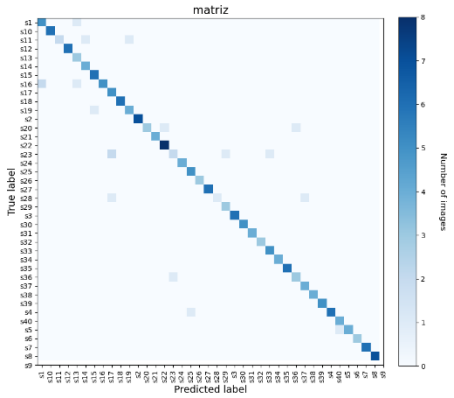
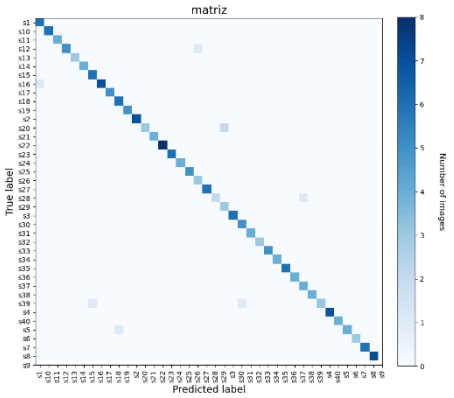
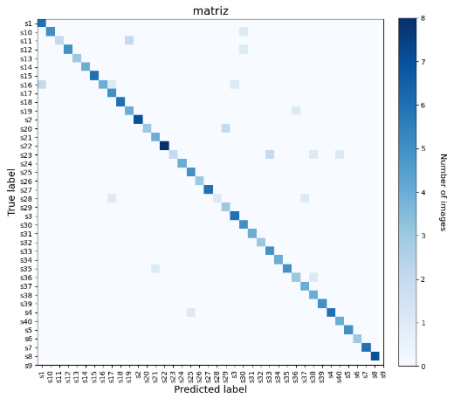
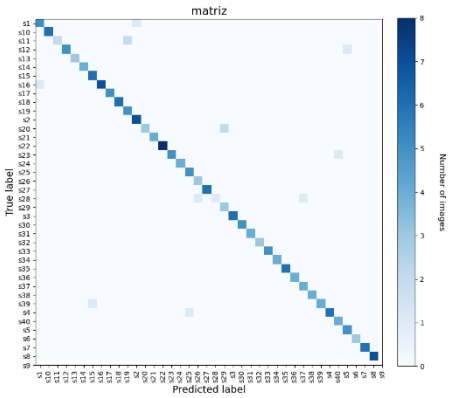
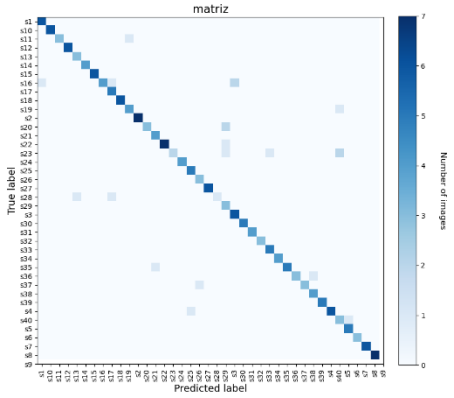
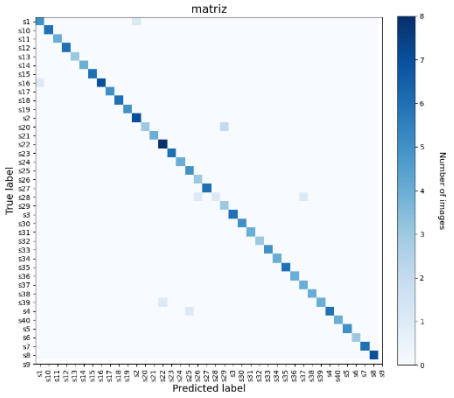
- 使用 sklearn.discriminant_analysis. LinearDiscriminantAnalysis 來進行 LDA

```
def dimensionality_reduction_LDA(n_components, X_train, y_train):
    print("Extracting the top %d fisherfaces from %d faces"% (n_components, X_train.shape[0]))
    pca = PCA(n_components=n_components).fit(X_train)
    lda = LDA().fit(pca.transform(X_train), y_train)

    return lda, pca
def train_text_transform_LDA(lda, pca, X_train, X_test):
    print("Start Transform")
    X_train_lda = lda.transform(pca.transform(X_train))
    X_test_lda = lda.transform(pca.transform(X_test))

    return X_train_lda, X_test_lda
```

二、 結果

- PCA 和 LDA 各個維度的混淆矩陣

	PCA	LDA
10		
20		

30	 matriz True label Predicted label Number of images	 matriz True label Predicted label Number of images
40	 matriz True label Predicted label Number of images	 matriz True label Predicted label Number of images
50	 matriz True label Predicted label Number of images	 matriz True label Predicted label Number of images

- 各個維度的識別率

	PCA	LDA
10	85.00%	93.00%
20	88.50%	96.50%
30	91.00%	96.00%
40	90.00%	94.00%
50	90.00%	96.00%

三、 討論結果

在實作此次作業的過程中發現降維的不同，會影響結果的表現。單純比較 PCA 及 LDA，會發現 LDA 整體表現都較為優異，結果為 LDA 的 20 維表現最好。

四、 總結

在此次作業當中學習到了許多新的程式技巧，如何讀取資料夾內的所有影像檔、切割資料、降維、建立模型以及畫出混淆矩陣。我個人覺得老師的課程作業相對地比較困難，但是學到的東西也比較多，需要多花時間與功夫下去做。