Assignment 2: Sketch Recognition

B1043043 周定毅 資工四

HOG+SVM

- 採用HOG(Histogram of Oriented Gradients/方向梯度直方圖)來提取圖像的特徵.
- 原因:
  1. 保留圖像中的局部形狀資訊
  2. 對光線與背景變化不敏感
  3. 不需要大量資料就能運作（不像 CNN 那樣仰賴大量標註圖片）。
- SVM（Support Vector Machine）:強大的監督式機器學習演算法，常用於分類任務。
  - 優點：對高維資料（例如圖像特徵）表現好。不容易過擬合（尤其使用線性核時）。僅依賴 support vectors，不受非邊界點影響。
  - 限制：在資料量非常大或特徵非常複雜時，訓練時間會變長。不適合處理多分類（需要做 One-vs-One 或 One-vs-Rest 擴展）。

導入模組並設定HOG參數

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.color import rgb2gray
from skimage.feature import hog
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# HOG 參數
hog_params = {
    'orientations': 9,
    'pixels_per_cell': (8, 8),
    'cells_per_block': (2, 2),
    'block_norm': 'L2-Hys'
}
```

參數名稱	意義說明
orientations=9	每個 cell 的方向梯度分成 9 個 bins（0°~180°）
pixels_per_cell=(8,8)	每個 cell 為 8x8 像素的區塊
cells_per_block=(2,2)	每個 block 包含 2x2 個 cells，用來做 normalization
block_norm='L2-Hys'	使用 L2-Hys 方法進行向量正規化，提高對亮度變化的穩定性

定義特徵擷取函數

```
# 特徵擷取函數
def load_and_extract_features(folder, label):
    features, labels = [], []
    for filename in os.listdir(folder):
        if filename.endswith('.bmp'):
            img = imread(os.path.join(folder, filename))

            # 自動處理是否轉灰階
            if img.ndim == 3:  # RGB 彩色圖像
                img = rgb2gray(img)
            elif img.ndim != 2:
                continue  # 跳過不是2D或3D圖的檔案

            feature = hog(img, **hog_params)
            features.append(feature)
            labels.append(label)
    return features, labels
```

透過讀取圖片,將RGB圖片轉成灰階,若不是 2D（灰階）或 3D（RGB）圖像，直接跳過。

使用 hog() 萃取特徵；收集每張圖的 HOG 特徵與其對應的標籤（例如：花＝0、人臉＝1）。

設定資料夾並載入訓練與測試資料

```
# 資料夾路徑
train_flower_path = '/content/TrainingData/Flower'
train_face_path   = '/content/TrainingData/Faces'
test_flower_path  = '/content/TestingData/Flower'
test_face_path    = '/content/TestingData/Faces'

# 載入訓練資料
X_train_f, y_train_f = load_and_extract_features(train_flower_path, 0)
X_train_h, y_train_h = load_and_extract_features(train_face_path, 1)
X_train = np.array(X_train_f + X_train_h)
y_train = np.array(y_train_f + y_train_h)

# 載入測試資料
X_test_f, y_test_f = load_and_extract_features(test_flower_path, 0)
X_test_h, y_test_h = load_and_extract_features(test_face_path, 1)
X_test = np.array(X_test_f + X_test_h)
y_test = np.array(y_test_f + y_test_h)
```

透過前面定義的 load_and_extract_features() 函數萃取 HOG 特徵；label=0 表示花，label=1 表示人臉。

將兩類資料合併，組成最終的 X_train/y_train 和 X_test/y_test。

建立SVM模型並進行驗證與評估

```
# 建立模型
model = SVC(kernel='rbf', C=1.0)

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# 顯示結果
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9333333333333333
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        15
           1       0.93      0.93      0.93        15

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30

Confusion Matrix:
 [[14  1]
 [ 1 14]]
```

kernel='rbf': 適用於非線性可分的資料，能夠處理較複雜的邊界。

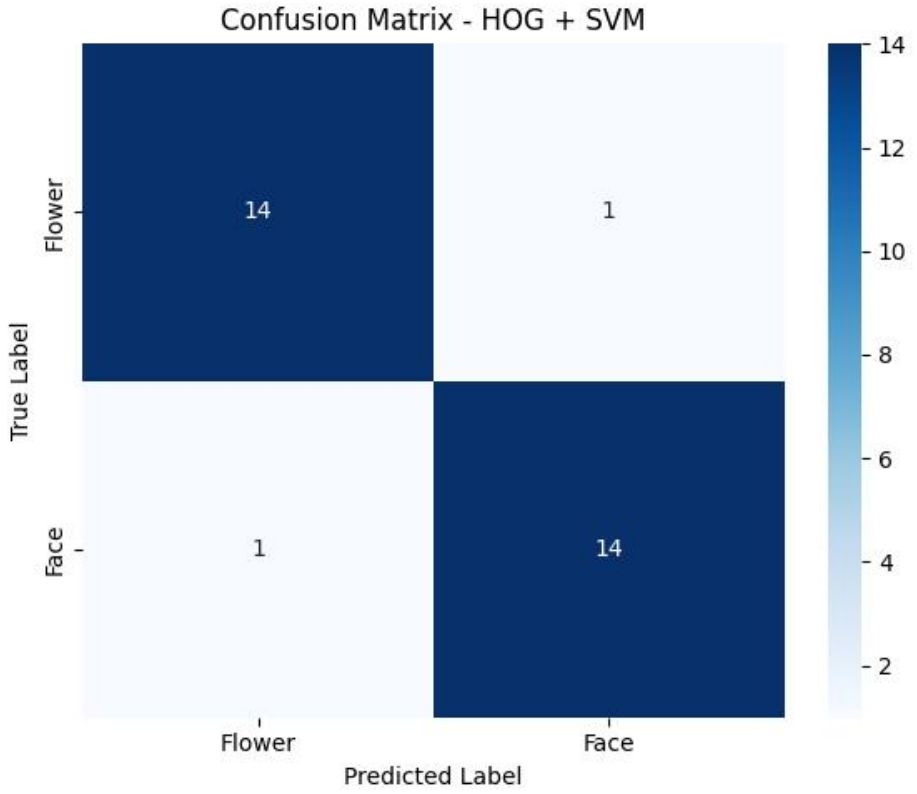
C=1.0:懲罰參數,用於平衡分類錯誤與間隔寬度.C 越大代表容忍錯誤越少,模型會嘗試更嚴格地分類訓練資料.

精確率 (Precision)：對於每一類，模型都能準確預測大約 93% 的正確類別，這意味著這個預測是非常可靠的。

召回率 (Recall)：模型能夠正確地找出 93% 的該類資料。這表明模型捕捉到了大部分的真實資料。

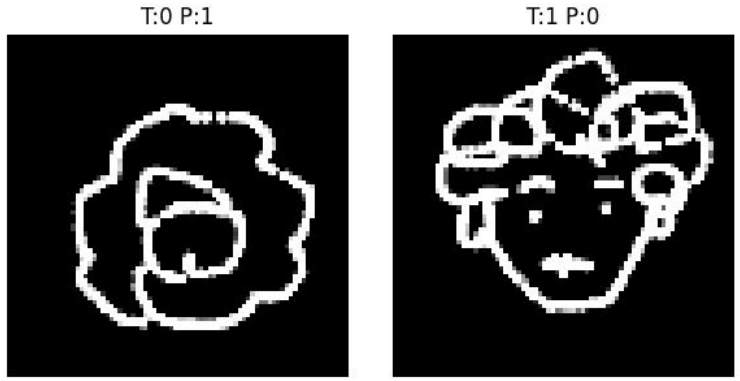
F1 分數 (F1-score)：由精確率與召回率的調和平均數來衡量，這是模型在兩者之間的平衡。在這裡，F1 分數為 0.93，表示模型在精確率和召回率之間找到了很好的平衡。

Confusion Matrix

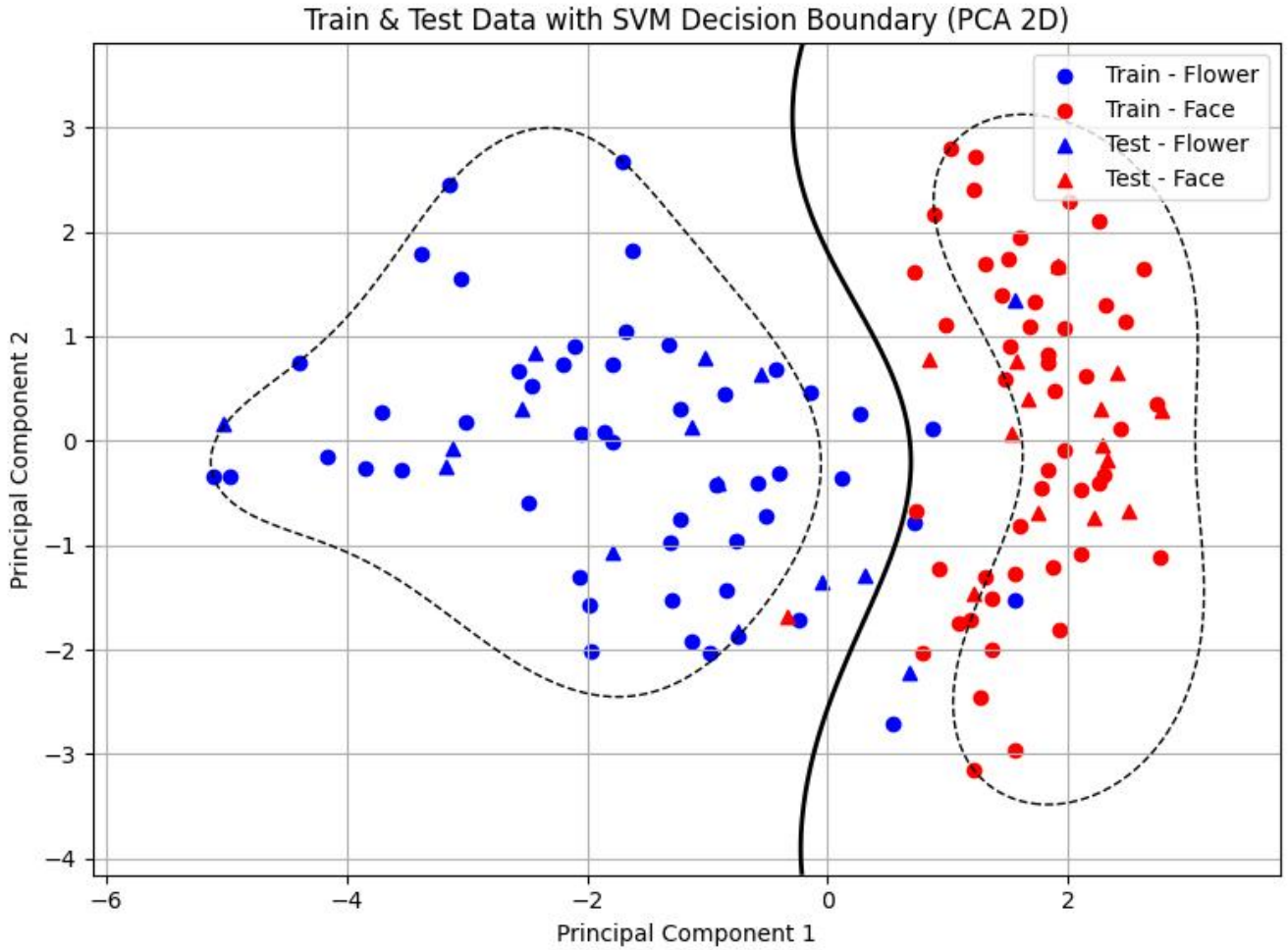


•模型在辨識花與臉時大致上準確，但有一些混淆：

- 花被誤判為臉（1 次）；
- 臉被誤判為花（1 次）。



視覺化所有的訓練與測試資料以及SVM的分界



決策邊界明確將兩個主成分空間中的類別分開。
訓練與測試資料整體分布一致，測試資料在邊界附近表現尚可。
有一些測試資料落在 margin甚至錯誤分類區域內，這是預期現象。

討論

SVM與Logistic Regression在此次分析中有何區別?

模型表現

- **SVM**：
  - 在許多情況下，SVM 在小樣本、高維數據的問題中能夠表現得非常好。由於它專注於最小化分類邊界的錯誤，並且對資料的噪聲有較高的魯棒性，SVM 能夠在較為複雜的資料集上提供較好的分類效果。
  - 但 SVM 訓練時間較長，尤其是當資料集變大時，計算需求也隨之增加。
- **邏輯回歸**：
  - 邏輯回歸在資料集較為簡單或接近線性可分時表現良好。它計算上較為高效且容易實現，但對於較為複雜的數據（尤其是高維、非線性可分的資料），邏輯回歸的表現通常不如 SVM。
  - 邏輯回歸的模型訓練速度快，對於資料量大的情況尤為優越。

```python
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
# SVM 模型
svm_model = SVC(kernel='rbf', C=1.0)
svm_model.fit(X_train, y_train)
y_svm_pred = svm_model.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, y_svm_pred))
print("Classification Report:\n", classification_report(y_test, y_svm_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_svm_pred))

# Logistic Regression 模型
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
y_log_pred = log_model.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_log_pred))
print("Classification Report:\n", classification_report(y_test, y_log_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_log_pred))
```

```
SVM Accuracy: 0.9333333333333333
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        15
           1       0.93      0.93      0.93        15

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30

Confusion Matrix:
 [[14  1]
 [ 1 14]]

Logistic Regression Accuracy: 0.9
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.87      0.90        15
           1       0.88      0.93      0.90        15

    accuracy                           0.90        30
   macro avg       0.90      0.90      0.90        30
weighted avg       0.90      0.90      0.90        30

Confusion Matrix:
 [[13  2]
 [ 1 14]]
```

SVM 和邏輯回歸模型的表現差異不大。兩者都達到了高準確率，並且在各類別的精確度和召回率上保持良好的平衡。SVM 的準確率略高於邏輯回歸（93.33% vs. 90%），這表明在這個特定問題中，SVM 使用 RBF 核表現稍微優越。這也可以解釋為 RBF 核更適合處理非線性問題，而邏輯回歸則對線性問題效果較好。

結論

- **模型表現**：在這個案例中，無論是 SVM 還是邏輯回歸，都表現得相當接近，準確度、精確度、召回率和 F1 分數基本非常接近。這顯示出這兩個模型在處理這個具體的問題上是有效的，且都能夠成功區分花卉和人臉兩個類別。
- **模型選擇**：由於兩個模型的表現非常相似，選擇哪一個模型可能取決於其他因素，如訓練時間、預測速度以及是否需要處理更複雜的非線性問題。SVM 通常在處理非線性問題時具有優勢，但在這種簡單的二分類問題中，邏輯回歸也能給出很好的結果。
- 總結來說，這兩個模型都能夠有效地處理這個分類問題，並且提供了相似的結果。選擇最適合的模型應該基於您的需求（如計算資源、模型可解釋性等）。