Image Clustering

B1043043 周定毅 資工四

特徵提取

- 採用ResNet50來提取圖片的特徵
  - ResNet50(Residual Network 50) 是一種深度卷積神經網路(CNN),包含 50 層,主要用於影像分類與特徵提取.
  - ResNet50 的完整架構包含捲積層+全連接層,但我們只需要卷積部分來提取特徵,並移除最後的全連接層,只輸出特徵向量.
  - 這樣的特徵向量可以表示影像的高層語義資訊(Semantic Features),例如形狀,紋理,物件特徵等.

設定資料夾與運算設備,加載 ResNet50 模型

```
folder_path = './images'
n_clusters = 3
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

設定圖片存放路徑,設定初始分群數量=3,設定使用 CUDA(GPU)或 CPU.

```
resnet = resnet50(weights=ResNet50_Weights.DEFAULT)
resnet = torch.nn.Sequential(*list(resnet.children())[:-1])
resnet.to(device)
resnet.eval()
```

使用 ResNet50 預訓練模型,移除最後的 Fully Connected(FC)分類層,保留特徵提取部分.
設定模型為 eval 模式,確保只進行特徵提取.

圖片前處理

```
transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

將.bmp圖片轉換至 PIL 格式,並將圖片調整到Resnet所要求的圖片大小,並將圖片轉換為 Tensor使用 ImageNet 的均值和標準差進行正規化.

特徵提取

```
image_features = []
image_paths = []

with torch.no_grad():
    for filename in os.listdir(folder_path):
        if filename.lower().endswith('.bmp'):
            path = os.path.join(folder_path, filename)
            img = cv2.imread(path)
            img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img_tensor = transform(img_rgb).unsqueeze(0).to(device)
            features = resnet(img_tensor).squeeze().cpu().numpy()
            image_features.append(features)
            image_paths.append(path)

image_features = np.array(image_features)
```

使用 torch.no_grad() 禁止梯度計算,加速推理.
使用 cv2.imread() 讀取圖片,並轉換為 RGB.
經過 transform 處理後,傳入 ResNet50 提取特徵.
特徵結果存入 image_features,圖片路徑存入 image_paths.
將特徵列表轉換為 NumPy 陣列,方便後續進行機器學習任務.

執行 KMeans 分群並顯示分群結果

```
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
labels = kmeans.fit_predict(image_features)
```

使用 KMeans 進行分群,並將結果儲存在 labels 中.

```
for cluster_id in range(n_clusters):
    plt.figure(figsize=(10, 2))
    plt.suptitle(f'Cluster {cluster_id}', fontsize=16)
    cluster_indices = np.where(labels == cluster_id)[0]

    for i, idx in enumerate(cluster_indices[:10]):  # Show first 10 images
        img = cv2.imread(image_paths[idx])
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(1, 10, i + 1)
        plt.imshow(img)
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

根據分群結果,選擇每個群組中的前10張圖片進行展示.

分群結果

Cluster 0

Cluster 1

Cluster 2

Silhouette Score 計算

```
scores = []
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    k_labels = kmeans.fit_predict(image_features)
    score = silhouette_score(image_features, k_labels)
    scores.append(score)
    print(f'k = {k}, Silhouette Score = {score:.4f}')
```

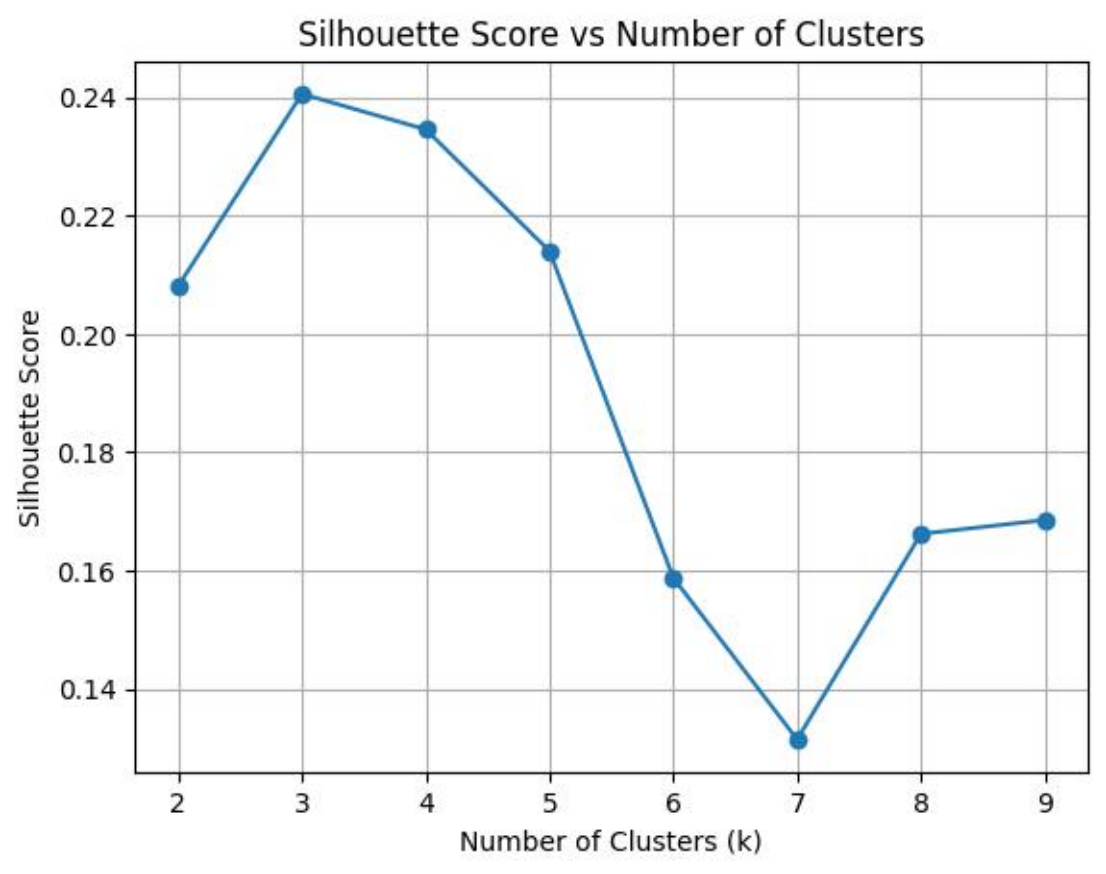
Silhouette Score 用於評估分群的質量,範圍在 -1 到 1 之間.分群效果越佳,分數越接近 1.

```
k = 2, Silhouette Score = 0.2081
k = 3, Silhouette Score = 0.2406
k = 4, Silhouette Score = 0.2346
k = 5, Silhouette Score = 0.2140
k = 6, Silhouette Score = 0.1588
k = 7, Silhouette Score = 0.1315
k = 8, Silhouette Score = 0.1662
k = 9, Silhouette Score = 0.1685
```

Silhouette Score 與群聚數量 (k) 的關係圖

```
plt.plot(range(2, 10), scores, marker='o')
plt.title("Silhouette Score vs Number of Clusters")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")
plt.grid(True)
plt.show()
```

繪製 Silhouette Score 與群聚數量 (k) 的關係圖



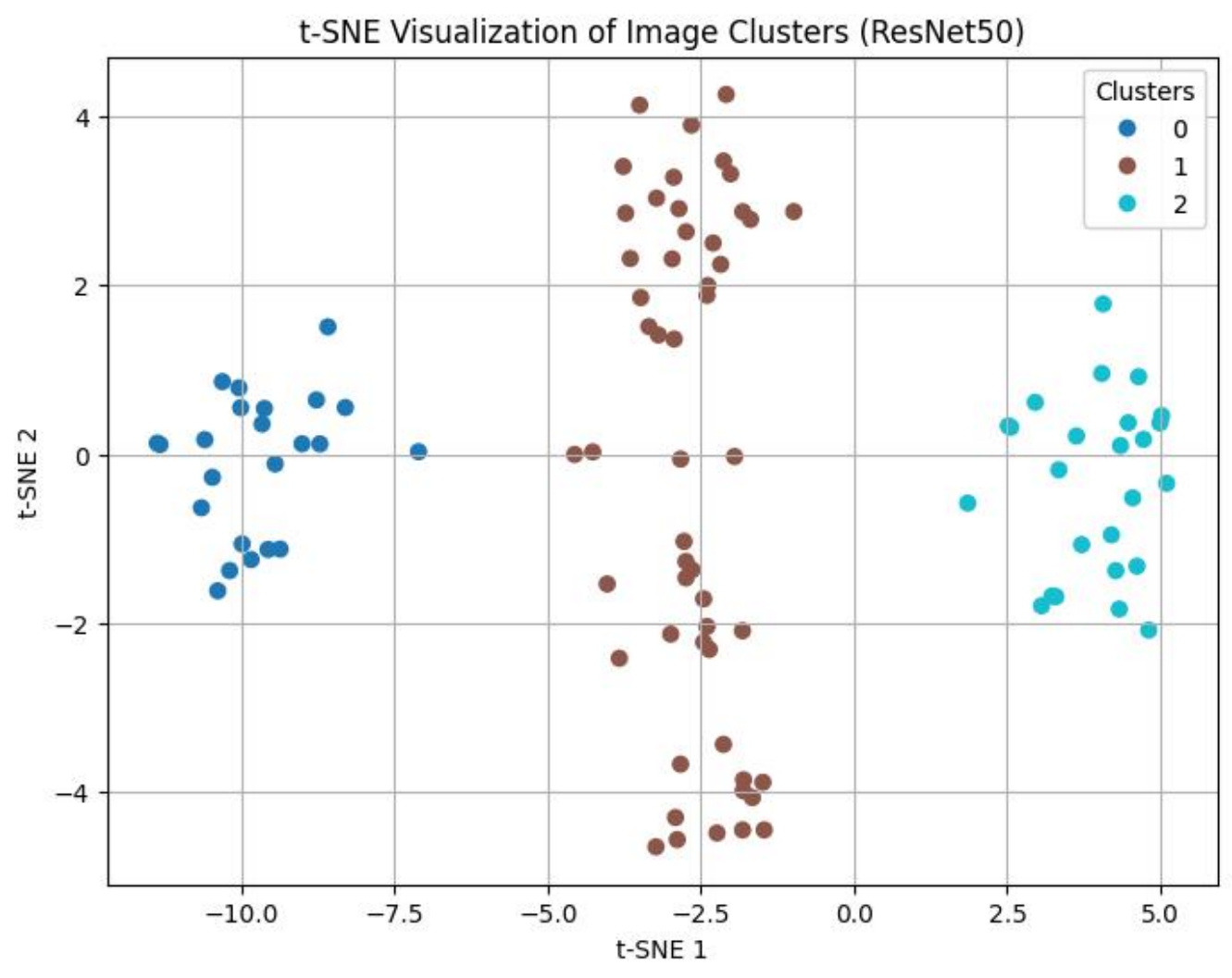
- 在第 3 個群聚 (k=3) 時,Silhouette Score 達到最高值 0.24,表示此時的分群效果最佳.
- 當 k 超過 3 時,Silhouette Score 開始下降,代表群聚之間的分隔性降低,分群效果變差.
- k=7 時的 Silhouette Score 最低,可能出現了過度分群或部分群聚混亂的情況.

t-SNE 降維視覺化

```
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
tsne_features = tsne.fit_transform(image_features)

plt.figure(figsize=(8, 6))
scatter = plt.scatter(tsne_features[:, 0], tsne_features[:, 1], c=labels, cmap='tab10')
legend1 = plt.legend(*scatter.legend_elements(), title="Clusters")
plt.gca().add_artist(legend1)
plt.title("t-SNE Visualization of Image Clusters (ResNet50)")
plt.xlabel("t-SNE 1")
plt.ylabel("t-SNE 2")
plt.grid(True)
plt.show()
```

由於ResNet50所輸出的結果為2048維,使用 t-SNE 將高維度的特徵壓縮至 2 維.
繪製分群結果,使用不同顏色標示.

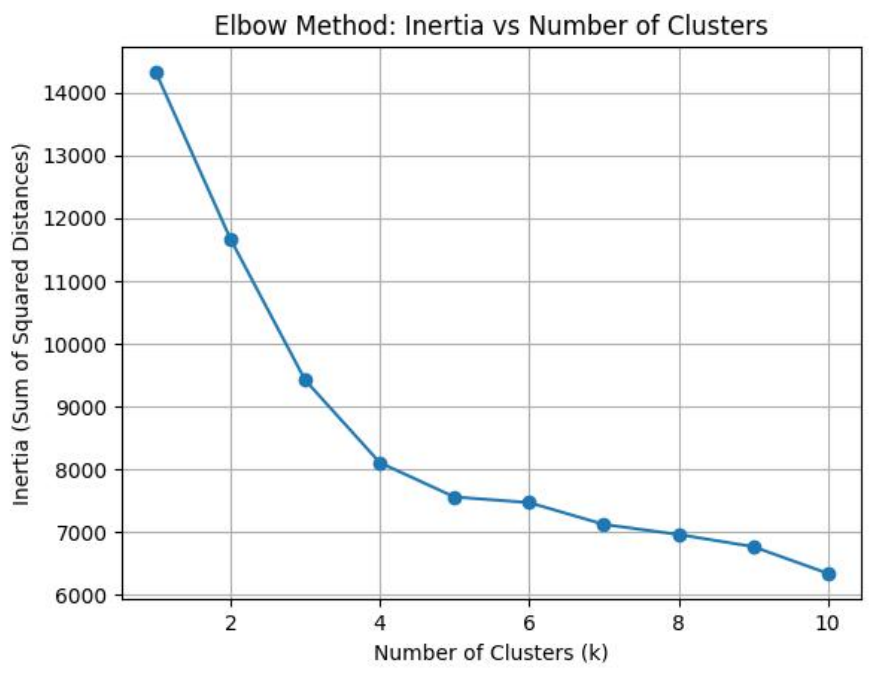


- t-SNE 將高維度的影像特徵壓縮至 2D,方便視覺化觀察分群結果.
- 圖中標示的三個群聚 (0, 1, 2) 彼此之間分離明顯,顯示 ResNet50 提取的影像特徵有良好的區分能力.
- 群聚 0,1,2 的分布大致呈現獨立的聚集區塊,代表模型有效地將相似特徵的影像歸為同一類.

Elbow Method 繪圖

```
inertia_values = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(image_features)
    inertia_values.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia_values, marker='o')
plt.title("Elbow Method: Inertia vs Number of Clusters")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia (Sum of Squared Distances)")
plt.grid(True)
plt.show()
```



Elbow Method 透過觀察 Inertia 值來選擇最佳的分群數量.
當曲線出現明顯的「肘部」位置時,即為最佳分群數.

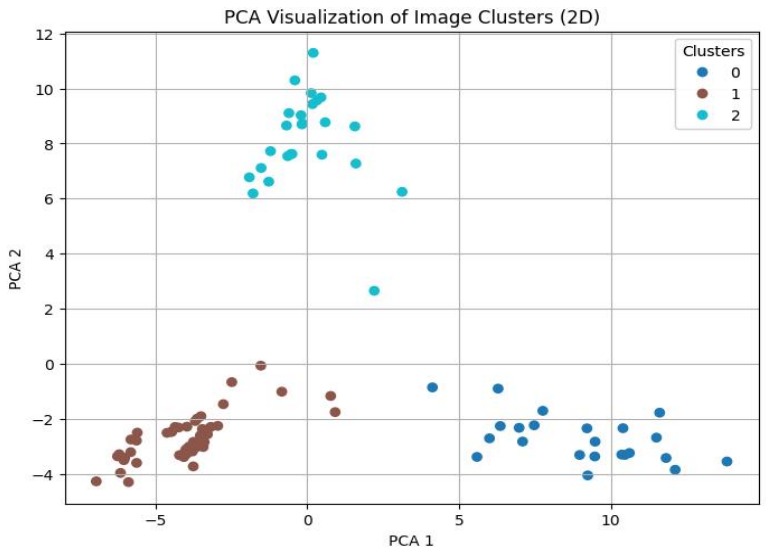
- Inertia 表示所有資料點到其所屬群聚中心的總距離,數值越低代表群聚效果越好.
- 在 k=3 處,Inertia 有明顯下降,但在 k=4 之後,下降幅度趨於平緩,形成所謂的 「肘部 (Elbow)」.
- 這表明增加群聚數量的邊際效益逐漸降低.

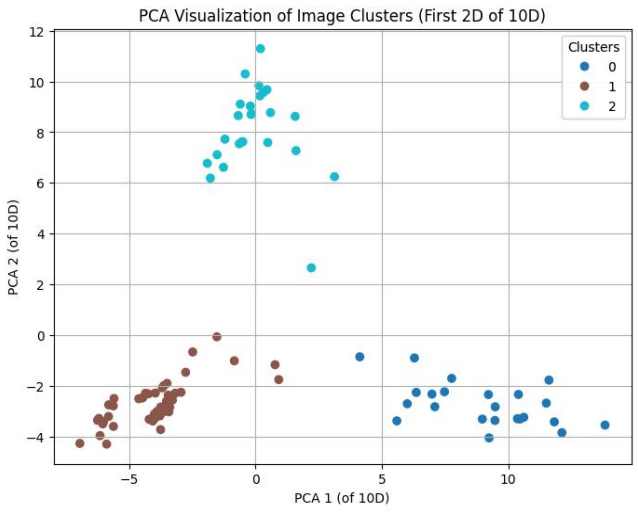
討論

- 降維是否會對分群結果產生影響?
  - t-SNE是為了可視化而設計的,使得其只能降到2-3維,故改用PCA(Principal Component Analysis)進行測試.
  - 透過計算2維和10維的可解釋變異比例(Explained Variance),可以得知維度對原始數據的貢獻程度.

```
Explained Variance Ratio for 2D PCA: [0.2070021  0.16148731]
Explained Variance Ratio for 10D PCA: [0.20700213 0.16148718 0.10647529 0.03689643 0.03317231 0.03047975
 0.023515   0.01912043 0.01763784 0.01435765]
```

2維保留了大約36.85%的變異,10維保留了63.70%的變異,但兩者的PCA分群圖差異不大.





可能的原因:

- 數據本身的低維特性或冗餘維度.
- PCA 的線性特性導致 2D 和 10D 的結果相似.
- KMeans 或 t-SNE 的侷限性,使得分群圖在高維和低維中表現相似.
- 數據的噪聲或均勻分佈掩蓋了高維結構.
- 視覺化的侷限性,導致無論降到 10D 還是 2D,最終的視覺化結果都是 2D 的.

結論

- 綜合 Silhouette Score,t-SNE 視覺化 和 Elbow Method 的結果,最佳的群聚數量為 k=3.這代表影像資料依據 ResNet50 提取的特徵,可以有效地分成三個明顯的群聚.
- Silhouette Score 接近 0,表示:
  - 分群有一定的合理性:群聚間有部分差異,系統能辨別主要的群集.
  - 邊界資料點較多:可能存在大量的資料點位於群集邊界,導致分群結果不夠明確.
  - 群集內部不夠緊密或群集之間距離不夠遠:這可能是由於資料的分布較為模糊,或選擇的特徵維度無法充分區分群集.