

# 期末專題報告

課程名稱：人工智慧

## 照片分類模型

開發語言、工具：Android studio Kotlin 、深度學習、

影像處理、資料庫設計

S0854045 資工四 曾筠惠

民國 110 年 6 月 19 日

# 內容

需求分析.....	3
設計說明.....	4
系統架構.....	7
開發流程.....	7
程式碼.....	12
成果.....	16
總結心得.....	19

# 需求分析

## 說明：

IG 上看到很多網美的版面都是同一個色調，例如：冷色調、暖色調等等，版面乾淨、整齊可以上傳照片，而這些照片是怎麼從手機相簿幾千張照片找出來的呢，要如何從幾千張照片找到那一張接近版面色調的照片呢？

常見 IG 色調		
		
冷色調	暖色調	白色調

此外我發現版面看起來會如此整齊原因有三點 分別是 1. 物件 2. 背景 3. 調色 可以利用同一色系物件(藍色包包)，背景(藍色大海)去達到統一色調的效果，再不行就是調色，將白色的雪調成偏藍色。

有哪些特徵		
		
物件	背景	調色

根據上述觀察，我希望可以利用人工智慧，做出可以分類色彩的模型，並結合 APP 實作介面。

# 設計說明

## 照片分類模型

**目標：**1. 可以分類照片色彩的模型

**用途：**1. 方便使用者整理照片 2. 社群網站版面/發文

主要分成兩方向探討：

### 1. 顏色偵測

**功能：**擷取圖片實際顏色

**作法：**使用 K-means 分群方法

### 2. 色系分類

**功能：**將圖片實際顏色歸類到我想分類的色系

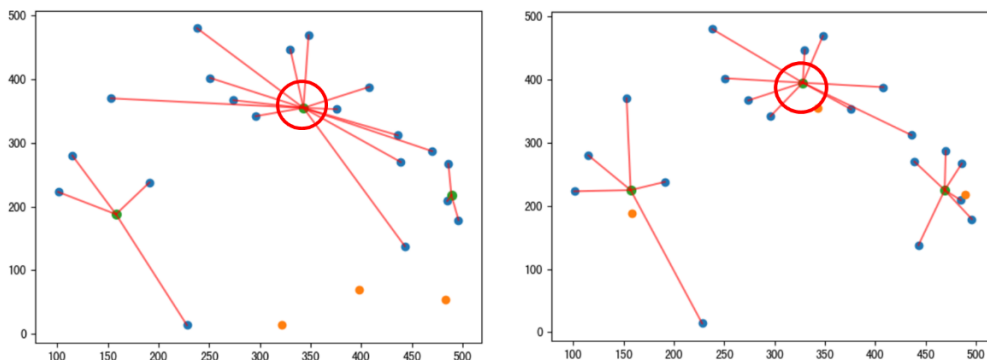
**作法：**使用 RGB threshold 將 RGB 色彩分類成 gray、warm、cool

三類

# 1. 顏色偵測

## ● K-means 分群步驟：

1. 先決定要分 k 組，並隨機選 k 個點做群集中心。
  2. 將每一個點分類到離自己最近的群集中心(直線距離)。
  3. 重新計算各組的群集中心(常用平均值)。
- 反覆 2、3 動作，直到群集不變，群集中心不動為止。



黃色是上一步群集中心的初始點，綠色為新的群集中心。

## ● 為什麼使用深度學習訓練？

### 1. 可處理高維度的數據

顏色由多個特徵或屬性組成，如 RGB（紅綠藍）值或 HSV（色相、飽和度、亮度）值。

### 2. 自動學習特徵

傳統的機器學習方法需要手動提取特徵，可能無法完全捕捉顏色的差異。深度學習模型可以學習到較佳的特徵表示，更好區分不同的顏色。

### 3. 處理非線性關係

顏色之間的關係是非線性的，傳統的機器學習方法難以捕捉到這種複雜的關係。

## ● 為什麼選擇 K-means 模型？

### 1. 簡單而直觀

K-means 是一種簡單且易於理解的聚類算法，可以將每個像素視為一個數據點，並使用 K-means 算法將像素分為 K 個不同的顏色集群。

### 2. 計算效率高

相對於某些複雜的深度學習算法，K-means 算法的計算效率相對較高，對大型圖像數據集處理來說很重要。

### 3. 可解釋性強(彈性高)

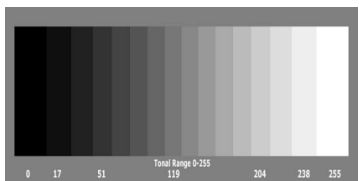


K-means 生成的結果易於解釋和理解，且可以調整 k 值，根據自己的需求對結果進行後續處理。

因為我的設計是一次只需分類一張照片，且延遲不能太久，所以選擇高效率且簡單的 K-means 模型。

## 2. 色系分類

我將色系分成三類 1. 灰階 2. 冷色調 3. 暖色調，並嘗試將顏色 RGB 值歸類到此三類。

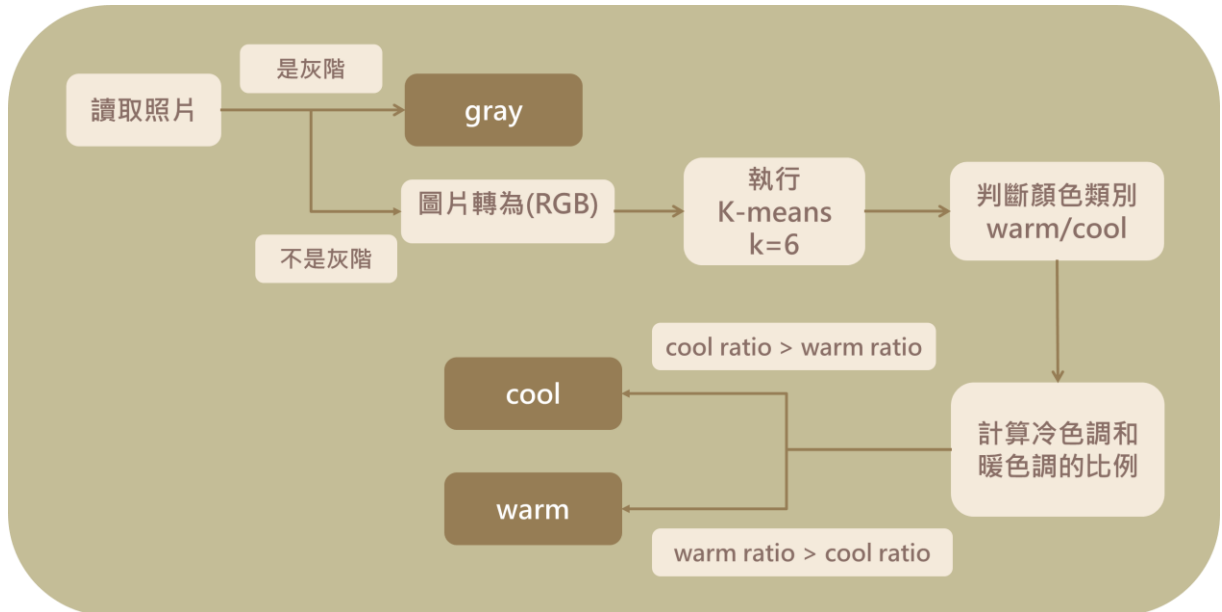
透過下表 RGB 觀察

灰階	冷色調	暖色調
		
R=G=B	(55, 64, 213) (64, 213, 104)	(213, 64, 64) (213, 153, 64)

觀察後發現

冷色 R 小 暖色 R 大，我根據這些特徵寫出判斷式

# 系統架構

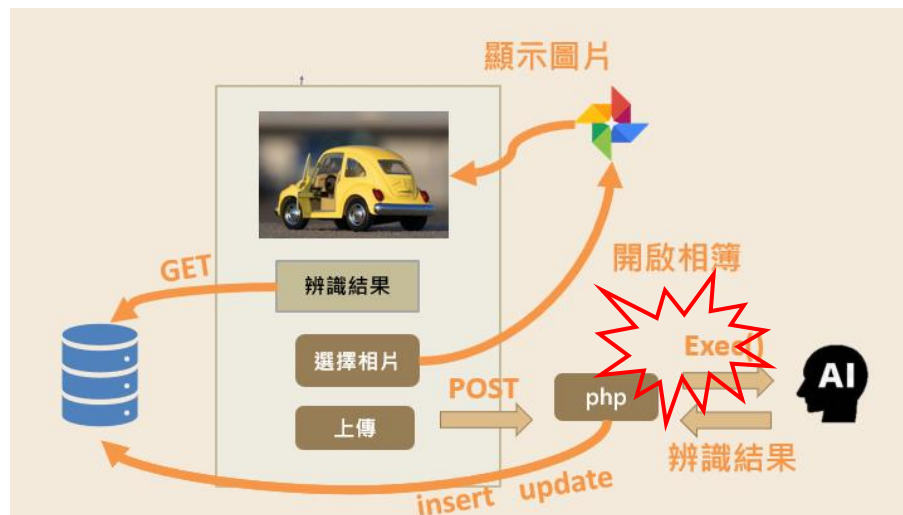


# 開發流程

## ● 與 APP 結合

APP 在接收到圖片後會呼叫 python 執行 K-means 分類

操作流程圖：



## ● K-means 分群

說明：

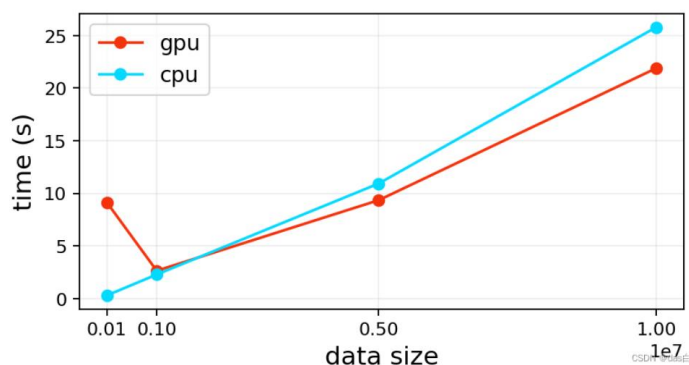
- 將圖片的像素數據（一維數組 RGB 值）作為輸入，為向量空間中的一個點。
- 將向量空間中的每個點分群，由使用者指定分成 k 群，便會有 k 個質心。（k=6）
- 顏色相近的點，在空間中的距離接近，會被分到同一質心，形成同一群。
- 最後重新計算 6 個群集，每一群集的質心位置（質心位置就是一組 RGB 位置），得到代表色與資料點數量，用圓餅圖呈現。

## ● K-means module 選擇

方法	硬體	比較
Scikit-learn	cpu	優點： 適合進行簡單分類任務 劣勢： 當數據量較大時，迭代速度相對較慢
網路上人家寫的 kmeans function	gpu	精度較差
kmeans-pytorch	gpu	能解決迭代速度問題 與 sklearn 相同的精度結果



CPU 和 GPU 執行 k-means module 比較表：



方法數量	512	1024	2048	16384	100000	1000000
CPU (sklearn)	0.39	0.16	0.25	3.12	28.91	278.90
GPU (kmeans-pytorch)	0.09	0.14	0.26	3.22	52.20	489.06

說明：

可以看到 GPU 在數量多時 因為要 CPU GPU 來回轉換所以顯得比較慢。

我在測試的時候也是因為 GPU 要切換所以很慢，且因為我需求只需要讀一張照片 CPU 還可以負荷，所以選擇 CPU。

## ● 灰階判斷

說明：

剛剛圖片有提到灰度圖中，RGB 三個通道的值是相等的，即  $R=G=B$ ，並且這些值的大小反映了灰度的深淺，從白色到黑色。

然而，僅僅通過檢查 RGB 圖像中的三個通道值是否相等是無法判斷區塊是否為灰度圖的。因為即使三個通道值不完全相等，但差異不大的情況下，對於人眼來說也可以視為灰度圖。

步驟：

- 獲取圖像區塊的像素值。
- 計算每個像素點的 RGB 通道平均值，即計算 R、G、B 的平均值。
- 將每個 R、G、B 的值與平均值進行比較，計算它們的差異。
- 判斷差異值是否小於某個閾值。如果差異值小於閾值(10)，則認為該區塊是灰度圖；否則，認為該區塊不是灰度圖。

程式碼：

```
#判斷圖像區塊是否為灰度圖
def checkGray(chip):

    # 將圖像區塊轉換為灰度圖
    chip_gray = cv2.cvtColor(chip, cv2.COLOR_BGR2GRAY)

    # 將圖像區塊的BGR通道分離為三個獨立的通道 (r、g、b)
    r, g, b = cv2.split(chip)

    # 將通道數值轉換為浮點型數值，以便進行計算
    r = r.astype(np.float32)
    g = g.astype(np.float32)
    b = b.astype(np.float32)

    # 獲取通道的寬度和高度
    s_w, s_h = r.shape[:2]

    # 計算每個通道的平均值作為灰度參考值
    x = (r + b + g) / 3

    # 計算每個通道與灰度參考值的差異
    r_gray = abs(r - x)
    g_gray = abs(g - x)
    b_gray = abs(b - x)

    # 計算每個通道差異值的總和除以區塊的像素數量，得到每個通道的差異值平均值
    r_sum = np.sum(r_gray) / (s_w * s_h)
    g_sum = np.sum(g_gray) / (s_w * s_h)
    b_sum = np.sum(b_gray) / (s_w * s_h)

    # 計算三個通道差異值平均值的平均值，得到灰度程度
    gray_degree = (r_sum + g_sum + b_sum) / 3

    # 如果灰度程度小於10，則判定為灰度圖像
    if gray_degree < 10:
        # print("Gray")
        return 1
    else:
        # print("NOT Gray")
        return 2
```

## ● 暖/冷色系

說明：

利用閾值 if...else...判斷屬於哪一個色系，且因為我的需求是想找出主要色系，所以乍看之下看不出是冷/暖的顏色，都會被歸類在 unknown。

步驟：

- 輸入 RGB 值
- 判斷 RGB 值是否大/小於某個閾值。如果差異值大/小於閾值，則認為該區塊是暖/冷色。

程式碼：

```
# 定義函式用於判斷顏色類別
def get_color_category(color):
    r, g, b = color

    # 設定閾值範圍
    cool_threshold = 60
    warm_threshold = 60

    # 判斷顏色類別
    if g - r > cool_threshold or r < cool_threshold :
        # print(" cool")
        return 'cool'
    elif r > warm_threshold :
        # print(" warm")
        return 'warm'
    else:
        # print(" unknown")
        return 'unknown'
```

# 程式碼

```
#CPU版本
# python final_project_cpu.py D:/Xampp/htdocs/app/images/1000.jpg
import sys
import os
import numpy as np
from sklearn.cluster import KMeans #CPU版本
import matplotlib.pyplot as plt
import cv2

num_colors = 6 # 指定顏色數量
```

函式:判斷冷暖色

```
# 定義函式用於判斷顏色類別
def get_color_category(color):
    r, g, b = color

    # 設定閾值範圍
    cool_threshold = 60
    warm_threshold = 60

    # 判斷顏色類別
    if g - r > cool_threshold or r < cool_threshold :
        # print(" cool")
        return 'cool'
    elif r > warm_threshold :
        # print(" warm")
        return 'warm'
    else:
        # print(" unknown")
        return 'unknown'
```

## 函式:判斷灰階

```
#判斷圖像區塊是否為灰度圖
def checkGray(chip):

    # 將圖像區塊轉換為灰度圖
    chip_gray = cv2.cvtColor(chip, cv2.COLOR_BGR2GRAY)

    # 將圖像區塊的BGR通道分離為三個獨立的通道 (r`g`b)
    r, g, b = cv2.split(chip)

    # 將通道數值轉換為浮點型數值，以便進行計算
    r = r.astype(np.float32)
    g = g.astype(np.float32)
    b = b.astype(np.float32)

    # 獲取通道的寬度和高度
    s_w, s_h = r.shape[:2]

    # 計算每個通道的平均值作為灰度參考值
    x = (r + b + g) / 3

    # 計算每個通道與灰度參考值的差異
    r_gray = abs(r - x)
    g_gray = abs(g - x)
    b_gray = abs(b - x)

    # 計算每個通道差異值的總和除以區塊的像素數量，得到每個通道的差異值平均值
    r_sum = np.sum(r_gray) / (s_w * s_h)
    g_sum = np.sum(g_gray) / (s_w * s_h)
    b_sum = np.sum(b_gray) / (s_w * s_h)

    # 計算三個通道差異值平均值的平均值，得到灰度程度
    gray_degree = (r_sum + g_sum + b_sum) / 3

    # 如果灰度程度小於10，則判定為灰度圖像
    if gray_degree < 10:
        # print("Gray")
        return 1
    else:
        # print ("NOT Gray")
        return 2
```

## 主程式：

由 php 利用 exec()函式 執行 python

### Php 執行python檔

```
$pythonScript = "C:/Users/90072/AIclass/final_project_cpu.py";  
$condaEnv = "base"; // Anaconda环境名称  
  
$pythonCmd = "conda run -n $condaEnv python $pythonScript $filename";  
$output = [];  
$returnValue = null;  
exec($pythonCmd, $output, $returnValue);
```

回傳值 回傳狀態

輸入

```
# 讀取單張照片  
# image_path = "D:/Xampp/htdocs/app/images/1004.jpg" # 替換為實際的照片路徑  
# 接收php 參數 'D:/Xampp/htdocs/saveimage/images/1000.jpg'  
  
# print("sys.argv 內容：" sys.argv)  
image_path = sys.argv[1]  
# 讀取圖片  
image = cv2.imread(image_path)  
  
# cv2.imshow('My Image', image)  
# cv2.waitKey(0)  
  
# 設置目標圖像尺寸  
target_width = 500 # 替換為你想要的寬度  
target_height = int(image.shape[0] * (target_width / image.shape[1]))  
  
# 降采样/調整圖像尺寸  
resized_image = cv2.resize(image, (target_width, target_height))
```

```
# 判斷圖片是否為灰度圖  
if checkGray(resized_image) == 1:  
    # 灰度圖像，直接分類為灰度  
    color_category = 'gray'  
    main_color = 'gray'  
else:  
    # 將圖片轉換為RGB色彩空間  
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  
  
    # 將圖片轉換為一維數組  
    pixels = image_rgb.reshape(-1, 3)  
    # 執行K-Means算法  
    kmeans = KMeans(n_clusters=num_colors)  
    kmeans.fit(pixels)  
    # 取得每個像素的標籤  
    labels = kmeans.labels_  
    # 調用KMeans對象的cluster_centers_屬性可以獲得聚類中心的RGB值，即每個顏色的代表值。  
    # 取得每個顏色的RGB值  
    colors = kmeans.cluster_centers_  
    # 計算每個色系的像素數量  
    counts = np.bincount(labels)  
    # 排序顏色和像素數量  
    sorted_colors = colors[np.argsort(counts)][::-1]  
    sorted_counts = np.sort(counts)[::-1]  
    # 計算冷色調和暖色調的比例加總  
    cool_sum = 0  
    warm_sum = 0
```

```

for i in range(num_colors):
    color = sorted_colors[i]
    # print(i, end=' ')
    # print(":", end=' ')
    category = get_color_category(color)
    if category == 'cool':
        cool_sum += sorted_counts[i]
    elif category == 'warm':
        warm_sum += sorted_counts[i]

# 計算冷色調和暖色調的比例
total_sum = cool_sum + warm_sum
if total_sum == 0 :
    main_color = 'gray'
else:
    cool_ratio = cool_sum / total_sum
    warm_ratio = warm_sum / total_sum

# 判斷最高比例的色調類別
if cool_ratio > warm_ratio:
    main_color = 'cool'
    main_ratio = cool_ratio
else:
    main_color = 'warm'
    main_ratio = warm_ratio

```

```

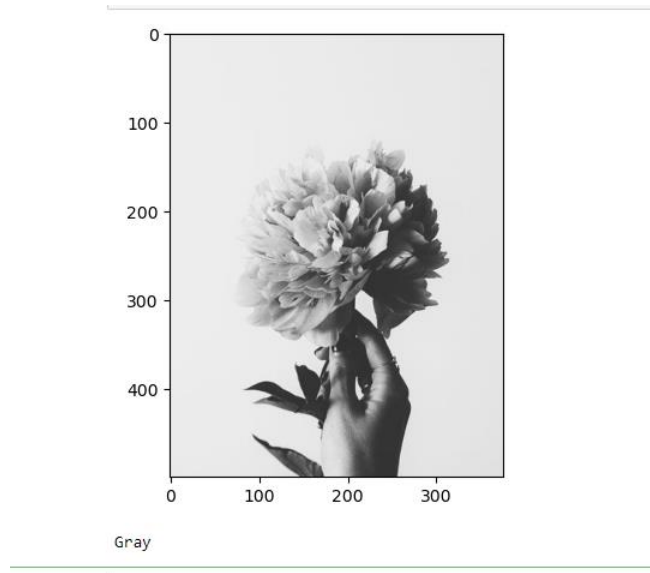
# 印出結果
print(f"")
print(f"圖片名稱：{image_path}")
print(f"冷色調比例：{cool_ratio * 100:.2f}%")
print(f"暖色調比例：{warm_ratio * 100:.2f}%")
print(f"主要色調：{main_color}")
print(f"主要色調比例：{main_ratio * 100:.2f}%")
# 繪製圓餅圖
labels = ['Cool', 'Warm', 'Gray']
ratios = [cool_ratio, warm_ratio, 1 - cool_ratio - warm_ratio]
colors = ['#66b3ff', '#ff9999', '#999999']
plt.pie(ratios, labels=labels, colors=colors, autopct='%1.1f%%')
plt.title('Color Composition')
plt.axis('equal')
plt.show()

print(main_color)

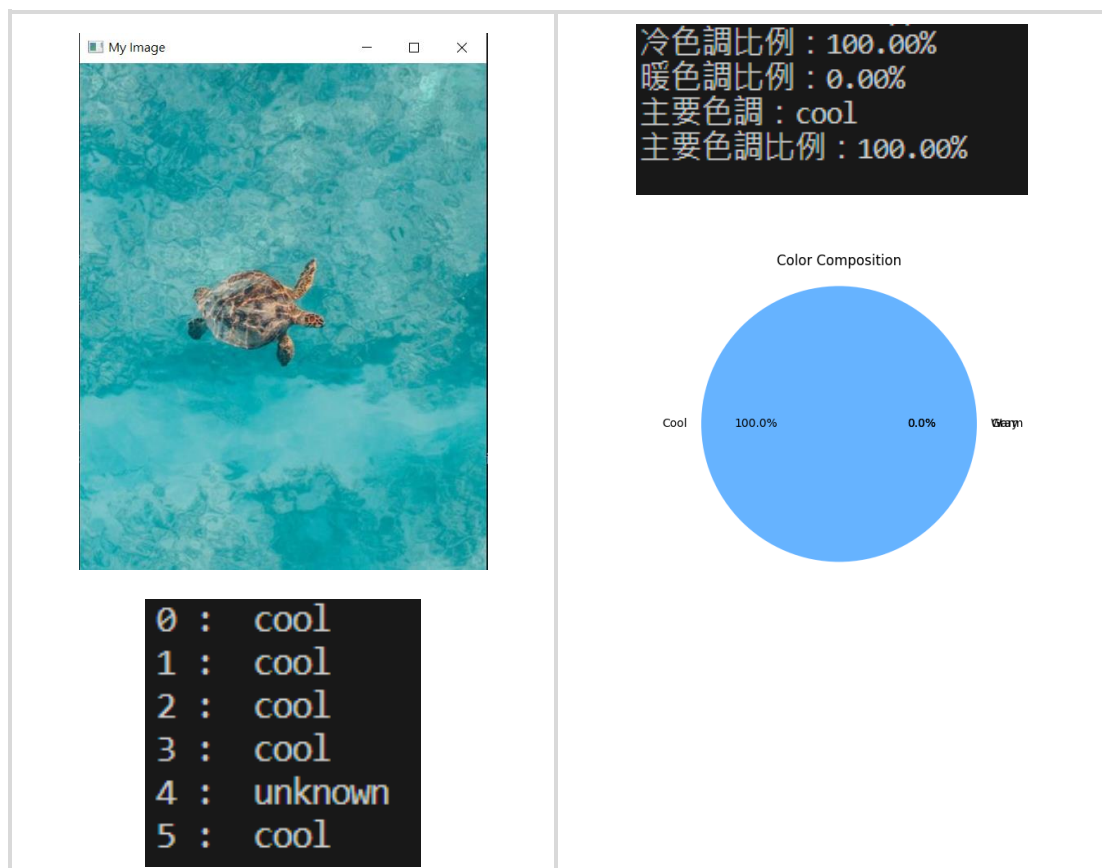
```

# 成果

## ● 灰階



## ● 冷色系

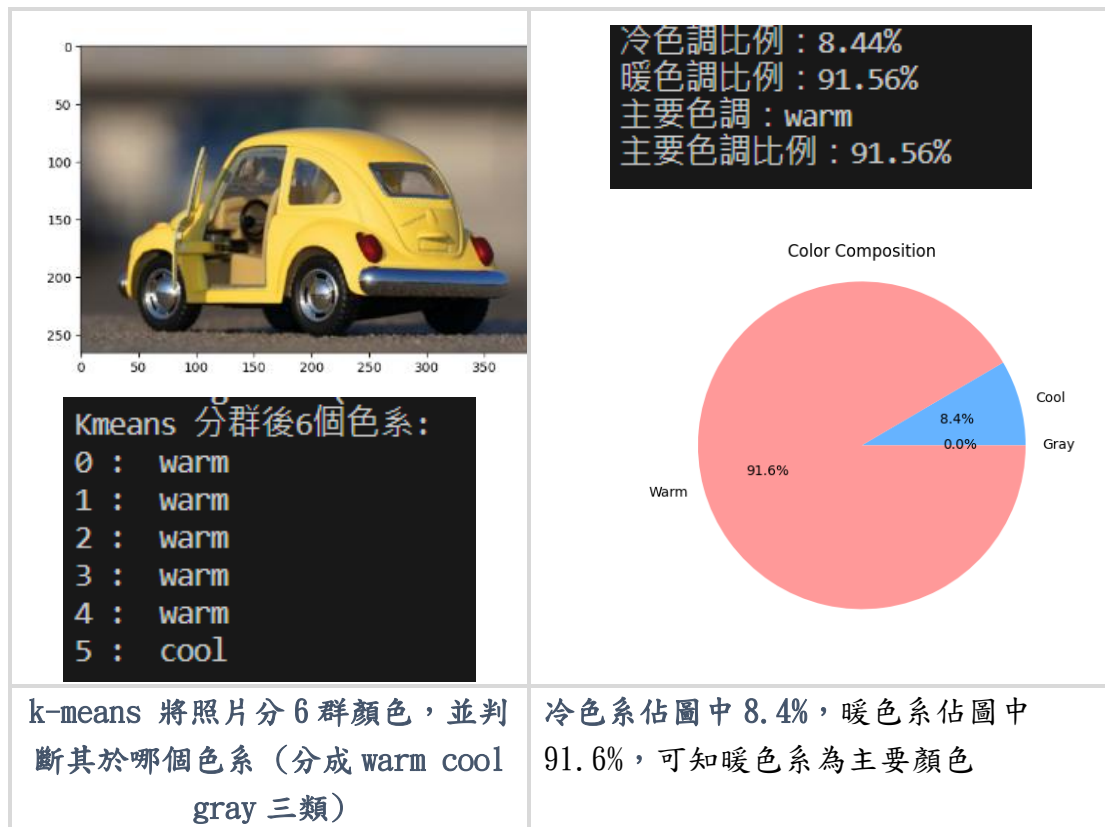




k-means 將照片分 6 群顏色，並判斷其於哪個色系（分成 warm cool gray 三類）

根據各色系在圖中比例計算主要色系

## ● 暖色系



- 結合 APP 介面實作成果

按下上傳後資料庫新增 a1001.jpg 此筆資料，再經果 AI 辨識將顏色結果 update 到資料庫

APP 介面

imgtitle	upload_date	content	color 顏色
a1001.jpg	2023-06-12 12:00:54	yy	cool
a1003.jpg	2023-06-12 12:21:56	456	cool
a1006.jpg	2023-06-08 06:29:07	黃色小車	warm
a1009.jpg	2023-06-08 17:20:03	橘色機車	warm

資料庫內容



yy

顯示辨識結果

選擇照片

圖片上傳成功

照片上傳至 server



yy

顯示辨識結果  
cool

選擇照片

上傳

辨識後從資料庫取得結果

## 總結心得

因為想要嘗試將 APP 和人工智慧結合，於是決定嘗試新題目，從一開始想分類底片，因為不同底片有不同色調，後來發現實作上有點困難，於是想到 IG 排版色調統一，從這個方向著手，因為是從 0 開始，也沒有人做過類似的題目，所以剛開始卡關很久，加上對 Android Studio 還不夠熟悉，還有硬體設備因素，自己每天在圖書館做都還是沒甚麼進度，想法跟怎麼做都很清楚，但當機、debug 就花了不少時間了，還好後來至少有做出大概的架構，細節防呆的部份需要在另外花時間完成。

這次是我第一次自己一個人開發一份專案，很有成就感，因為以前都是小組做專題，所以 debug 或是構想都可以一起討論，但這次所有事情都要自己來，變成依賴網路資料、Chat GPT，再將得到的資訊過濾，因為我發現 Chat GPT 回答不一定是正確的。

這次專題我學到如何尋找題目，並將構想修飾轉換成可以實作的題目，學會如何下查詢指令，都對以後研究所如何找論文題目很有幫助，老師在過程中也抽空幫我不少忙，給我一些提點，很謝謝老師，也祝老師在彰師教書可以順順利利。