第三堂課：影像辨識

顏色直方圖 + 最近鄰（few-shot，適合少樣本）

辨識 = 特徵擷取（把圖變一串數字）+ 比對（跟範本比誰最近）

註冊幾張範本 → 上傳新圖就認出是誰。從第二堂課的「平均色」升級成「真的會認」

淡江大學電機工程學系　AI 機器學習實驗室　·　2026-07

這堂課做什麼？

【導論】從第二堂課的「平均色」升級成「真的會認」

- **第二堂課只算「平均色」——太粗，只能說「偏紅、偏綠」，不能真的分辨是什麼**
- **第三堂課：讓板子真的「認出」上傳的圖是哪一類（先註冊範本，再比對）**

比較	第二堂課	第三堂課（本課）
特徵	平均色 3 個數字	顏色直方圖 24 個數字
做的事	偏哪色 / 亮暗	認出「這是哪一類」
怎麼判	比大小	跟範本算距離、找最近

資料流完全沿用第二堂課（上傳 → mmap 讀 → 分析），只換「分析」那塊 = 複習又進階。

辨識到底是什麼？ = 特徵 + 比對

【觀念】幾乎所有辨識都是這兩步

① **特徵擷取**

把圖變成「一串數字」（例如顏色直方圖）——抓住它的特色

② **比對**

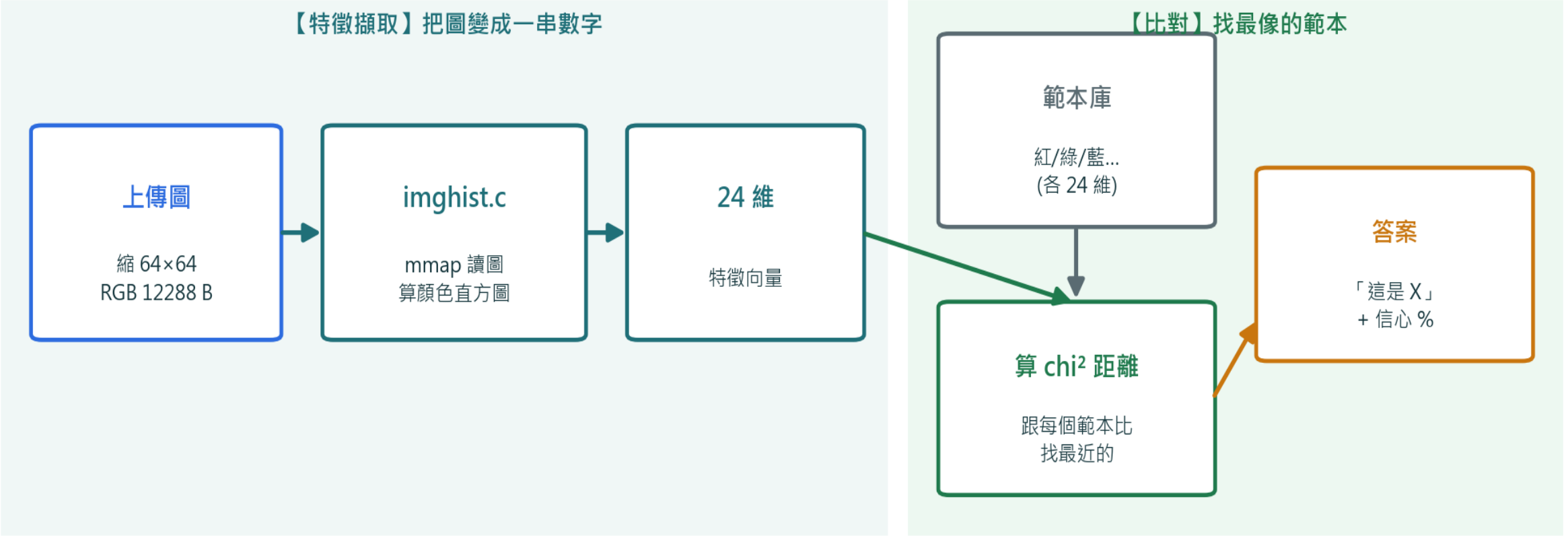
拿這串數字，跟「已知範本」的數字比——誰最接近，就判是誰

比喻：認人 = 記特徵（臉型、髮色、身高）+ 比對（跟你認識的人比誰最像）。機器辨識一模一樣。

換更強的「特徵」或「比對法」就更準——今天用最好懂的：顏色直方圖 + 最近鄰。

辨識流程總覽

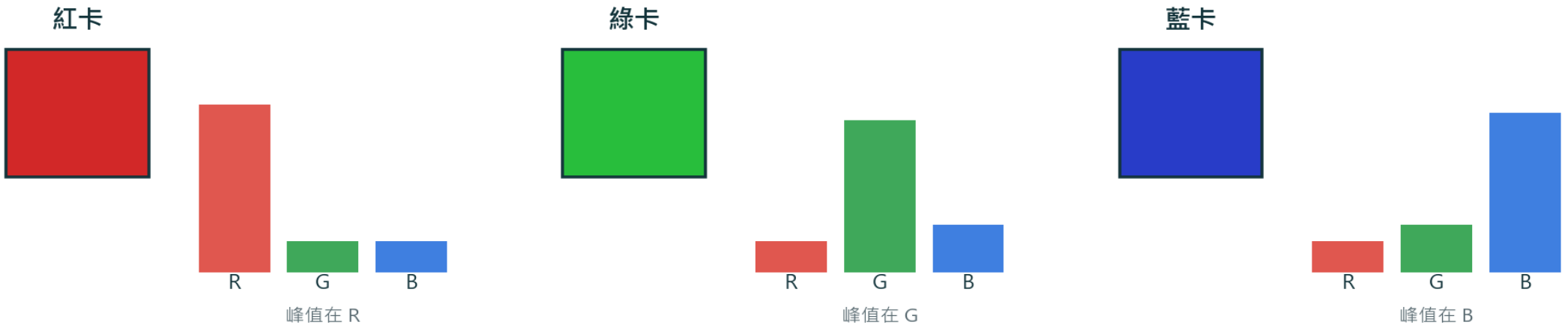
【全景】上傳圖 → 算特徵 → 跟範本比 → 答案（接下來逐步拆解）



把圖變成「顏色直方圖」（特徵）

STEP 1 / 3

辨識流程　第 1 步 / 共 3 步　·　特徵擷取 · 板子 mmap 讀圖後算



同一套特徵(顏色直方圖)：紅/綠/藍各自峰值在不同通道 → 分得開；新圖落在誰附近就判誰。

- 顏色直方圖 = 把每個通道(R/G/B)的顏色分佈「數成幾格」：這張圖有多少偏暗紅、多少偏亮紅...
- **每通道切 8 格 × 3 通道 = 24 個數字（正規化）= 這張圖的「特徵向量」**
- 紅/綠/藍的峰值在不同通道 → 特徵分得開 → 才認得出來

程式碼：imghist.c（mmap 算直方圖）

辨識流程　第 1 步 / 共 3 步　·　跟第二堂課同一套 mmap，只是改算直方圖

板子 ARM · imghist.c (C)

```
double hr[8]={0}, hg[8]={0}, hb[8]={0};   // 三通道各 8 格
for (long i = 0; i < 4096; i++) {         // 走過全部像素
    softbin(hr, buf[i*3+0]);   // R 落到哪格(軟性分格)
    softbin(hg, buf[i*3+1]);   // G
    softbin(hb, buf[i*3+2]);   // B
}
// 每格 ÷ 像素數(正規化) → 印出 24 個數字 = 特徵向量
```

- buf[] 從哪來：mmap 把 upload.rgb 映射成陣列（第二堂課學過，這裡原封不動）
- 輸出 24 維直方圖 + 平均色，交給 server.py 去比對（板子 python3 沒 numpy，C 算最省）

細節：軟性分格（一個真踩過的坑）

辨識流程　第 1 步 / 共 3 步　·　為什麼「硬分格」會把藍認成綠

- **坑：一開始用「硬分格」——像素值直接除以 32 取整數落到某一格**
- 問題：190 落在第 5 格、200 落在第 6 格 → 一點色差就「跨格」→ 藍色範本(200) 跟藍色新圖(190) 對不上 → 被誤判成綠！

```
// 軟性分格：依位置「加權分到相鄰兩格」，色差就不會突然跨格
void softbin(double *h, int v) {
    double p = v / 32.0;          // 0..255 → 0..7.97 連續位置
    int b0 = (int)p, b1 = (b0<7)? b0+1 : 7;
    h[b0] += 1 - (p - b0);        // 近的那格拿多一點
    h[b1] += (p - b0);            // 遠的那格拿少一點
}
```

改成軟性分格後：紅/綠/藍 6 種色調的新圖全部認對。這就是「特徵要穩」的重要性。

比對：跟每個範本比，找最近的

STEP 2 / 3

辨識流程　第 2 步 / 共 3 步　·　最近鄰 (Nearest Neighbor) + chi² 距離

- **新圖算出特徵後，跟「範本庫」裡每一張範本的特徵算「距離」——距離越小 = 越像**
- 距離用 chi²（卡方）：對直方圖比較穩：

```
距離 chi² = Σ (a - b)² / (a + b)      ← a=新圖某格, b=範本某格, 24 格加總
相似度    = 1 / (1 + 距離)            ← 距離越小、相似度越高
信心 %    = 這類相似度 ÷ 所有類相似度總和 × 100%
```

- **最近的那類 = 答案；信心 = 它比其他類「明顯」多少**
- 同一類可註冊多張範本（few-shot）→ 取最近的那張算距離，更耐色差/光線

程式碼：server.py（註冊 / 辨識端點）

辨識流程　第 2 步 / 共 3 步　·　純迴圈算距離，板子沒 numpy/json 也行

```
# 註冊：算這張圖的特徵，存進範本庫檔案
POST /register?name=紅卡  →  feat = imghist(upload.rgb)
                             templates.txt 加一行「紅卡 <24個數字>」

# 辨識：跟每個範本算 chi² 距離，找最近
POST /recognize  →  hist = imghist(upload.rgb)
  for name, h in 範本庫:
      d = Σ (hist - h)² / (hist + h)      # chi² 距離
      best[name] = min(該類目前最小距離, d)
  答案 = 相似度最高的那類；回 JSON（名字 + 信心 + 各類分數）
```

- **端點：/register（註冊一張）、/recognize（辨識）、/templates（看有哪些類）、/reset（清空）**
- 範本庫就是一個純文字檔 templates.txt（每行一張範本）——簡單、看得懂、好備份

找最近 → 算信心 → 回答；網頁兩模式

STEP 3 / 3

辨識流程　第 3 步 / 共 3 步　·　電腦端顯示：這是 X，信心 Y%

① 註冊範本　② 辨識

- ① 註冊：打個名字（紅卡/綠卡/...）→ 選圖 → 存成那一類的範本（1~10 張都行）
- ② 辨識：上傳新圖 → 板子算特徵、比對 → 回最像的類

板子回傳： `{"name":"紅卡","conf":61,"scores":[{"name":"紅卡","pct":61},{"name":"藍卡","pct":21},...]}`

- **網頁顯示：大字「🔎 這是【紅卡】信心 61%」+ 每類的分數長條 + 這張圖的直方圖**

整條路：上傳 → mmap 讀 → 算直方圖(特徵) → chi² 找最近(比對) → 回答。全部在板子上跑。

實測：註冊紅綠藍 → 認 6 種新色調

【實測】色調都跟註冊時不同，全部認對

上傳的新圖 (RGB)	認成	信心	各類分數
偏紅 (200,55,50)	紅卡	58%	紅58 綠21 藍21
偏綠 (55,180,70)	綠卡	59%	綠59 藍22 紅19
偏藍 (50,70,190)	藍卡	61%	藍61 綠22 紅18
深藍 (30,40,150)	藍卡	38%	藍38 紅32 綠30
橘紅 (220,90,40)	紅卡	52%	紅52 綠25 藍23
黃綠 (150,200,40)	綠卡	41%	綠41 紅34 藍24

6/6 全對。清楚的色 ~58-61%，模糊的色（深藍/黃綠）分數較低但仍判對。

為什麼用「簡單法」？——少樣本的智慧

【工程實務】真專案每類樣本很少，簡單法反而贏

- **真專案的限制：每一類可能只有 ≤10 張樣本（拍不到那麼多）**
- 深度 CNN 要幾千張才訓得動 → 樣本這麼少，硬訓只會過擬合、認不準
- **few-shot（少樣本）策略：用「顏色直方圖 + 最近鄰」這種簡單特徵法**
- **實證（真專案）：同樣 3 張樣本，深度 CNN 只對 1/3，顏色比對法 3/3**
- 教訓：資料少的時候，簡單、穩、可解釋的方法 > 複雜模型

這堂課的方法不是「玩具」——它正是少樣本影像辨識實務上的正確起手式。

檔案清單 & 怎麼佈署上板子才能用

【工程實務】最快：在 board_部署包\ 跑 部署.ps1 一鍵完成

檔案	跑在哪	做什麼
`imghist.c` ★	板子 ARM·C	核心：mmap 讀圖 → 算 24 維顏色直方圖（特徵）
`server.py`	板子 ARM·py3	註冊/辨識端點：純迴圈算 chi² 最近鄰、範本庫存檔
`recognize.html`	瀏覽器	網頁：①註冊範本 ②辨識（分數條 + 直方圖）

1 **一鍵部署**

在 board_部署包\ 跑： `powershell -File 部署.ps1 -Ip 板子IP`

2 **它自動做**

傳 `imghist.c/server.py/recognize.html` → 編譯 `imghist` → 重啟服務

3 **開始玩**

手機開 `http://板子IP:8080/recognize.html` → 註冊幾類 → 辨識

提醒：Project A（第二堂課）的 / 也還在，兩個 demo 共用同一個板子 server，共存不衝突。

總結 & 怎麼接到真系統

第三堂課一頁記住

- **學會：辨識 = 特徵擷取（顏色直方圖）+ 比對（chi² 最近鄰）；註冊範本 → 認新圖**
- 全部在板子上跑，沿用第二堂課的 mmap，few-shot 適合真專案的少樣本限制

要換的	現在（第三堂課）	更接近真系統
影像來源	電腦上傳的圖	FPGA 從 CIS 擷取、寫進 DDR3（前段，之後聯合課）
特徵	顏色直方圖 24 維	加 HSV / 邊緣 / 紋理，或換小型 CNN
比對	chi² 最近鄰	加信心門檻（太低回「不確定」）、多範本平均

先能認（本課已做到）→ 之後把來源換 FPGA、特徵/比對再加強，就一步步逼近真辨識系統。