

# 影像辨識應用於無人商店

## IMAGE RECOGNITION APPLIED IN UNMANNED STORES

---

**Key Words:**

Face Recognition, Object Detection, YoloV8,  
Google Cloud Platform, Edge Computing

STAT 4B 劉宸宇

- 1 Face Recognition
- 2 Object Detection
- 3 Deployment
- 4 Conclusion

# Face Recognition

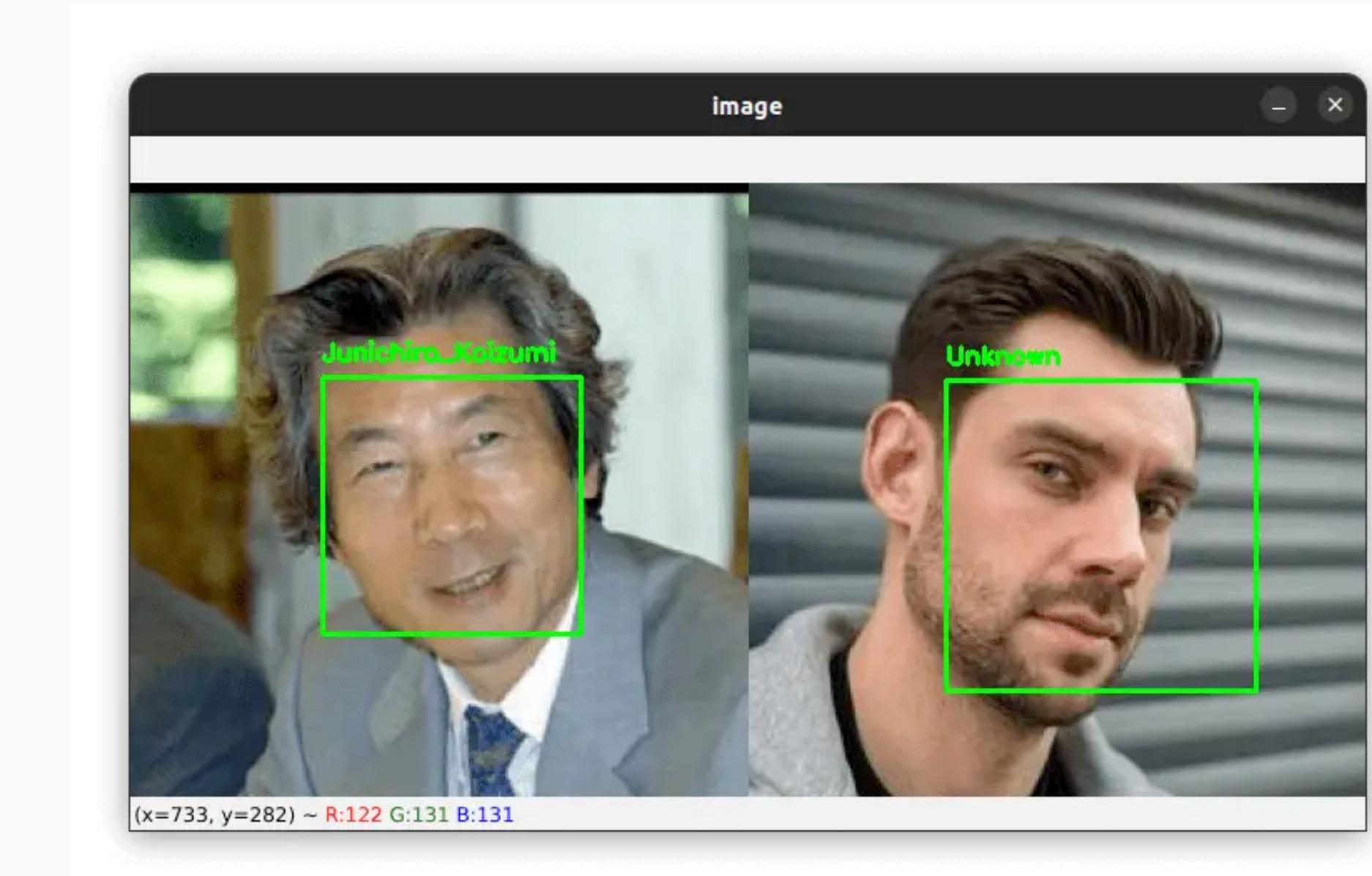
---

# Face Recognition Library

**face\_recognition** 是一個基於 Python 的人臉識別套件，利用 Dlib、NumPy 和 Pillow 等庫實現了簡單的人臉檢測和識別功能。

臉部辨識功能分為四個流程：

1. 尋找臉部位置
2. 踐部位置置中
3. 轉換人臉特徵編碼
4. 計算各人臉特徵編碼相似程度



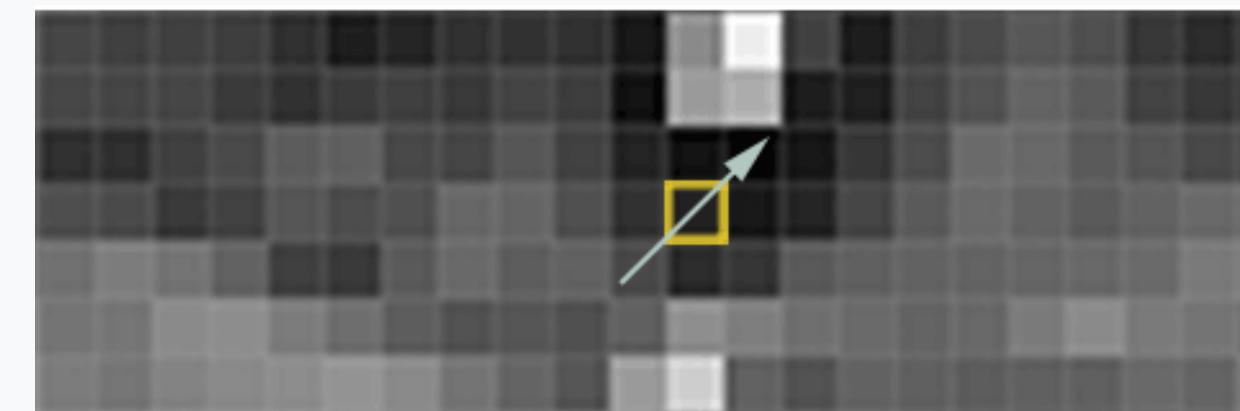
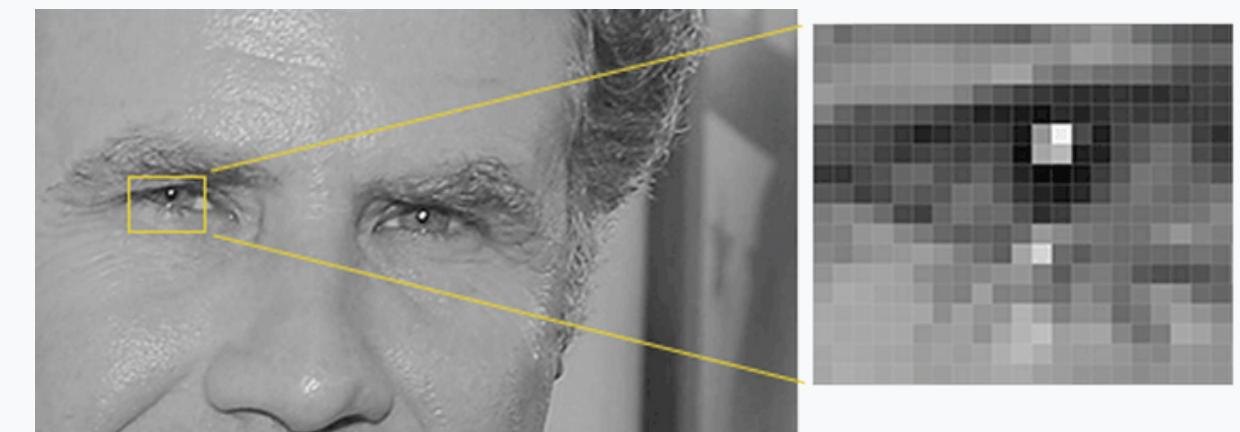
# Recognition Workflow - Face Location

## HOG (Histogram of Oriented Gradients)

臉部辨識第一階段會先透過HOG(Histogram of Oriented Gradients)方法，找出圖片輪廓，主要實現流程分為

1. 照片轉為灰階顯示
2. 圖像區塊劃分
3. 透過箭頭取代每一區塊影像，箭頭方向為每一區塊周圍亮度變為最黑的方向

將上述步驟對於圖片的每一個區塊進行處理後，圖片每一區塊會由不同方向箭頭所代表。



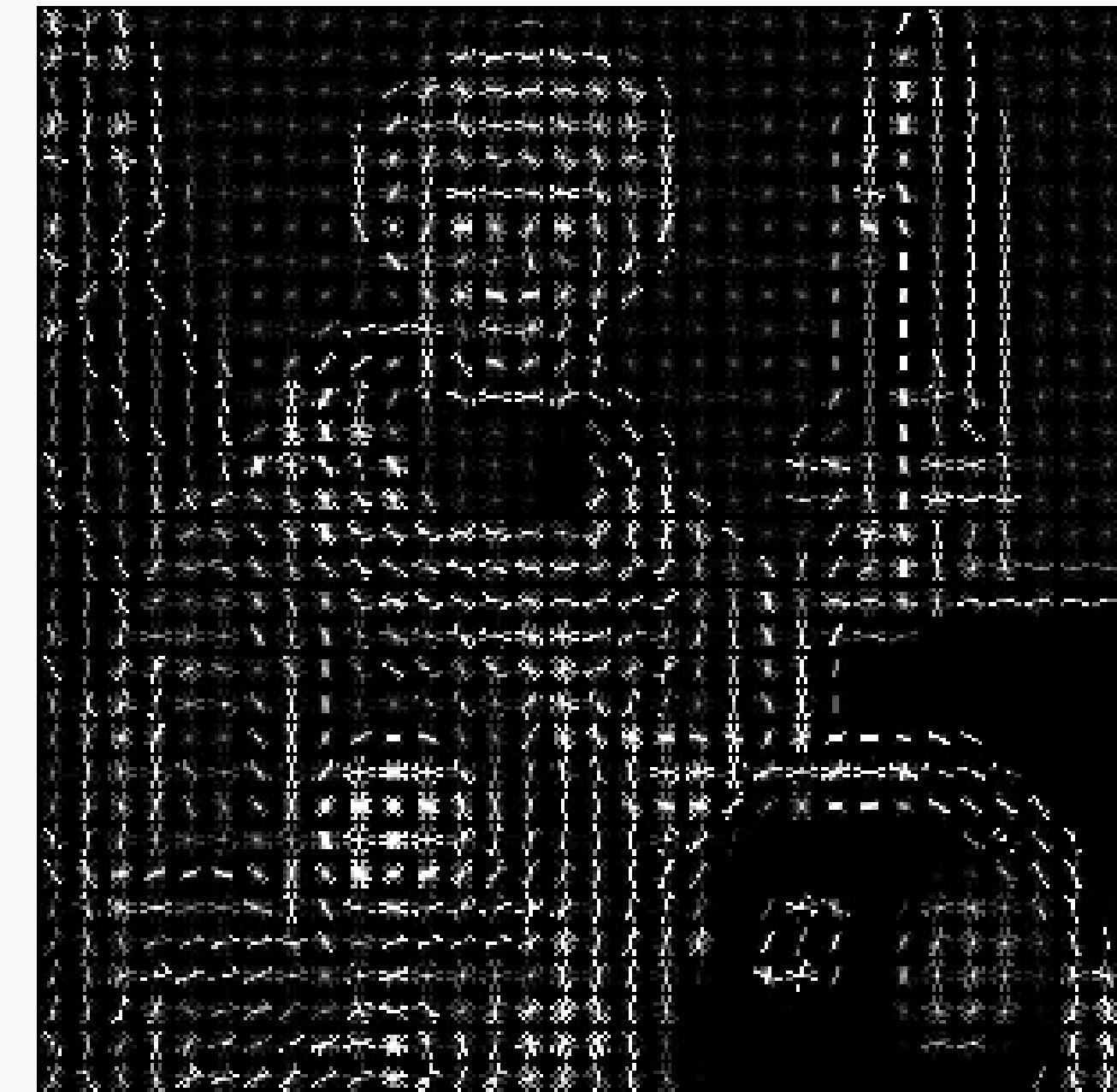
# Recognition Workflow - Face Location

這樣的方法能夠避免圖片的光線、顏色不同所導致的差異，在實際流程上，並不會對每一個像素都轉換，會先將圖片分為 $16 * 16$  區塊再進行轉換。

Input image

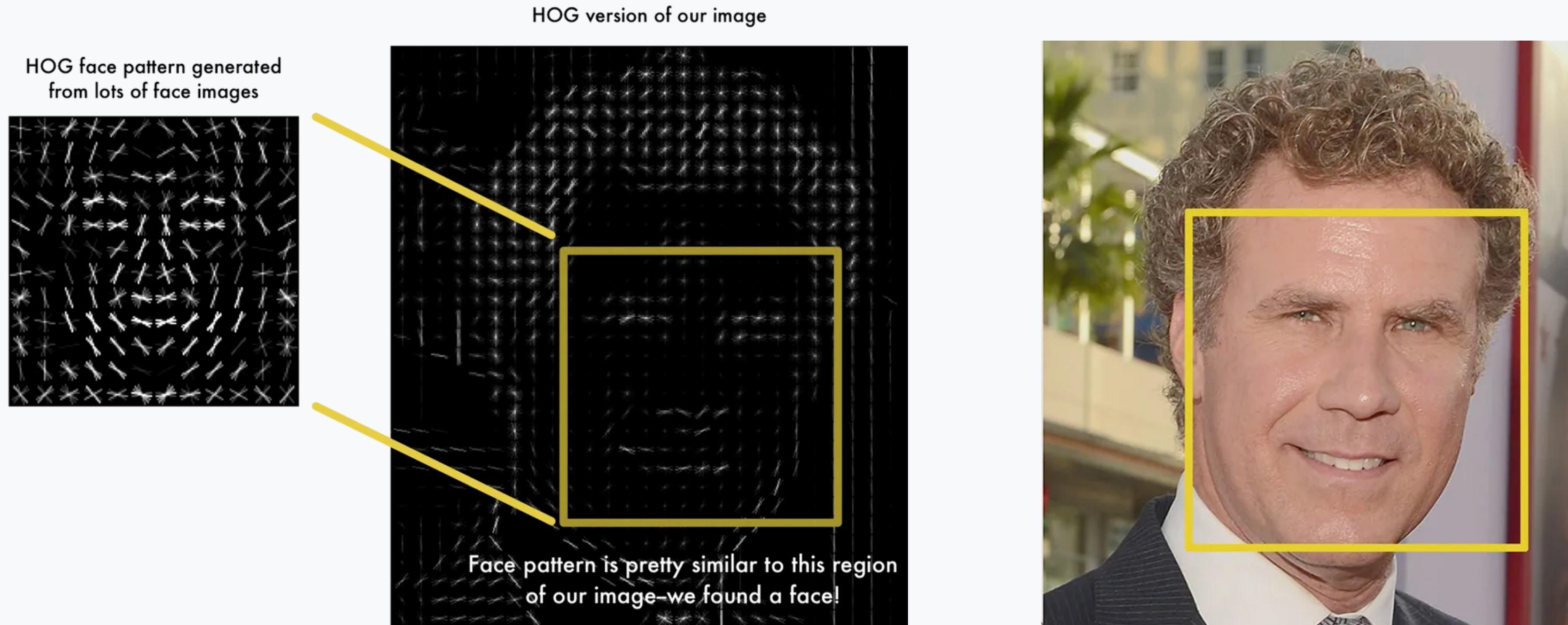


Histogram of Oriented Gradients



# Recognition Workflow - Face Location

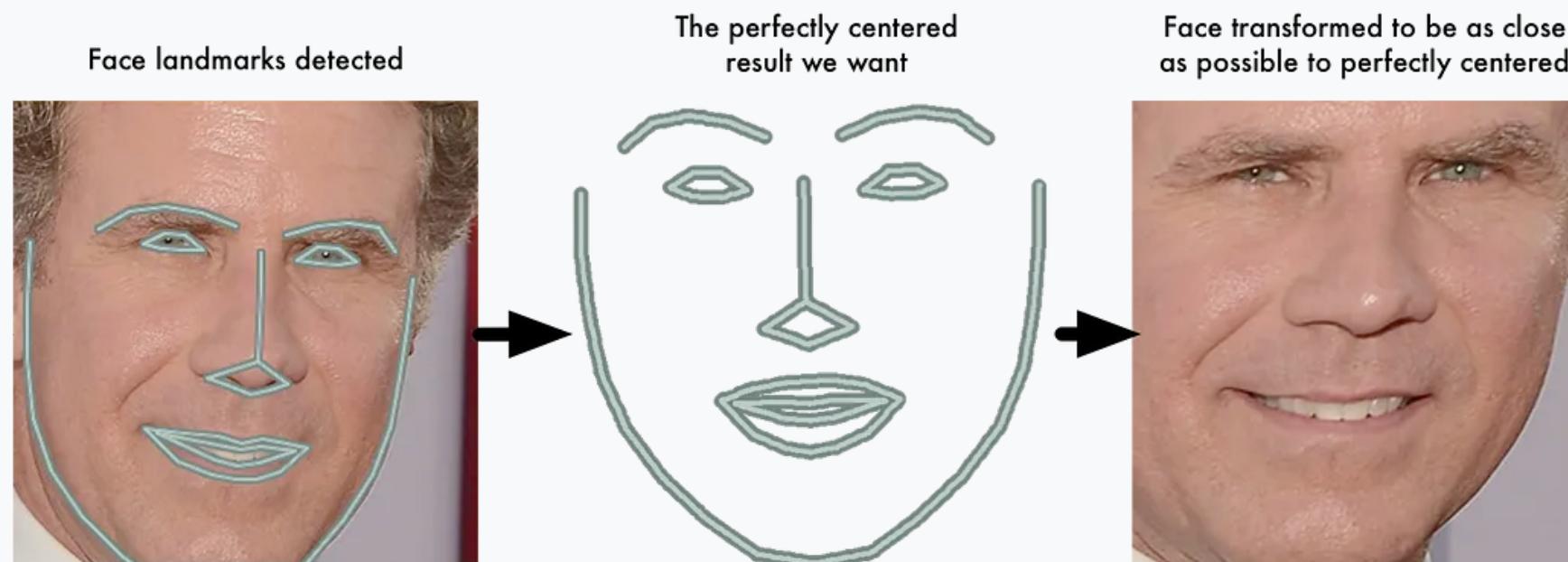
將所得到的**HOG結果和HOG face pattern**（透過大量臉部HOG圖像所得臉部輪廓）進行比較，找出該**HOG圖片臉部位置**，並將位置投影回輸入照片，即可得到該圖片的臉部圖片



# Recognition Workflow - Face Encoding

## 臉部位置：

將臉部位置找到後，由於角度的不同對於辨識結果會有影響，透過臉部68個基準點做仿射轉換，將臉部圖片轉至正面，並進行臉部編碼。



## 臉部編碼 (Face Encoding) :

將人臉轉換為數學向量或編碼，以進行比對、識別，核心思想是捕捉人臉圖像中的特徵，轉換為具體的數值，使得不同人臉之間差異能被量化。



# Recognition Workflow - Face Encoding

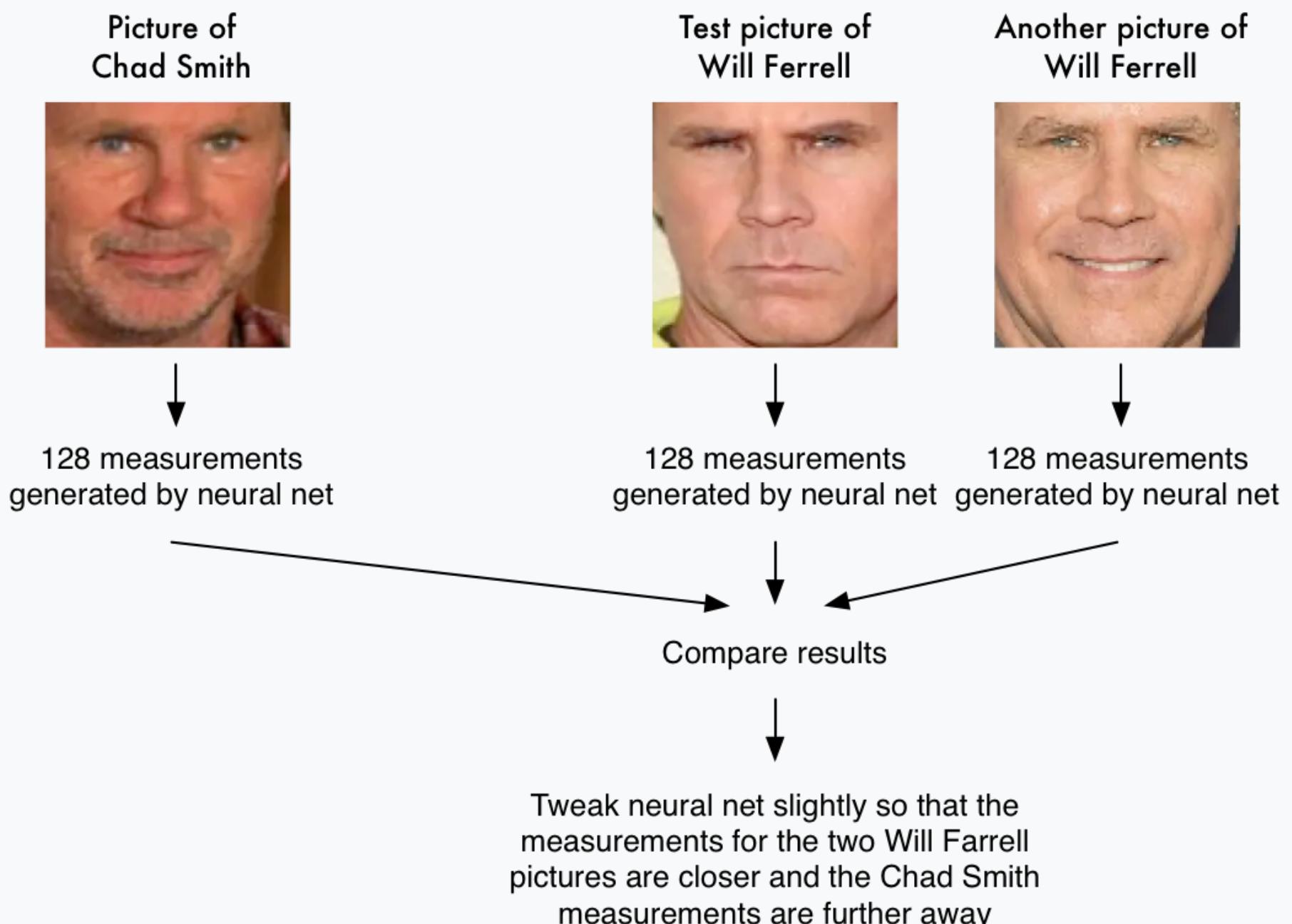
## Encoding 計算：

透過 Pretrained CNN 進行轉換，將圖片輸入進這個 CNN 後輸出該圖 Encoding

## Training Step：

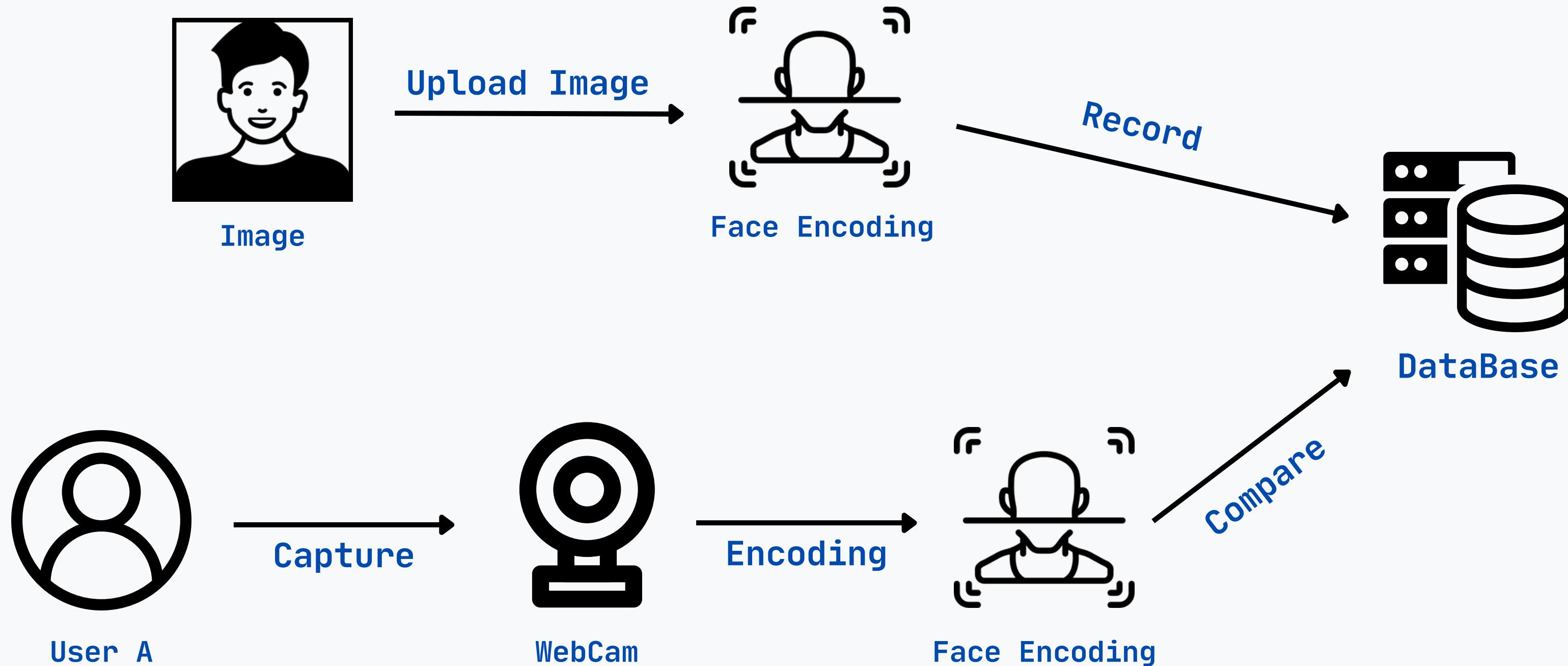
每次迭代都會分別輸入同一人兩張不同照片以及一張其他人的圖片，目的要盡量讓同一人不同照片所得 Output 有較近的距離，且跟另一人照片 Output 有較遠的距離

### A single 'triplet' training step:



# Face Recognition

## Adding User's Image

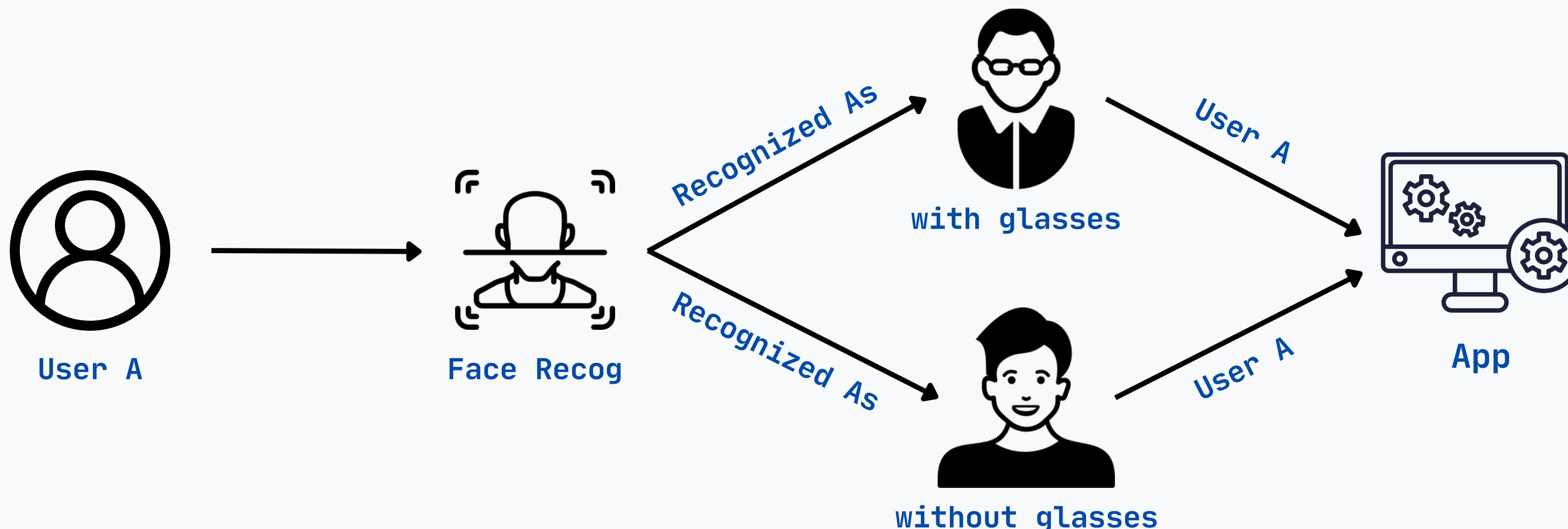


## Real Time Face Recognizing

# Improve Face Recognition

每個用戶僅有一張照片進行比較，可能因為角度，眼鏡等不同，影響辨識準確率。因此每位用戶可以提供數張不同場景的照片進行比較，避免這樣的問題。

例如，若用戶提供有無配戴眼鏡兩種照片，不論辨識時是否配戴眼鏡，都有對應到的辨識結果，而兩種結果對於資料庫都是同一用戶資料。



# Object Detection

---

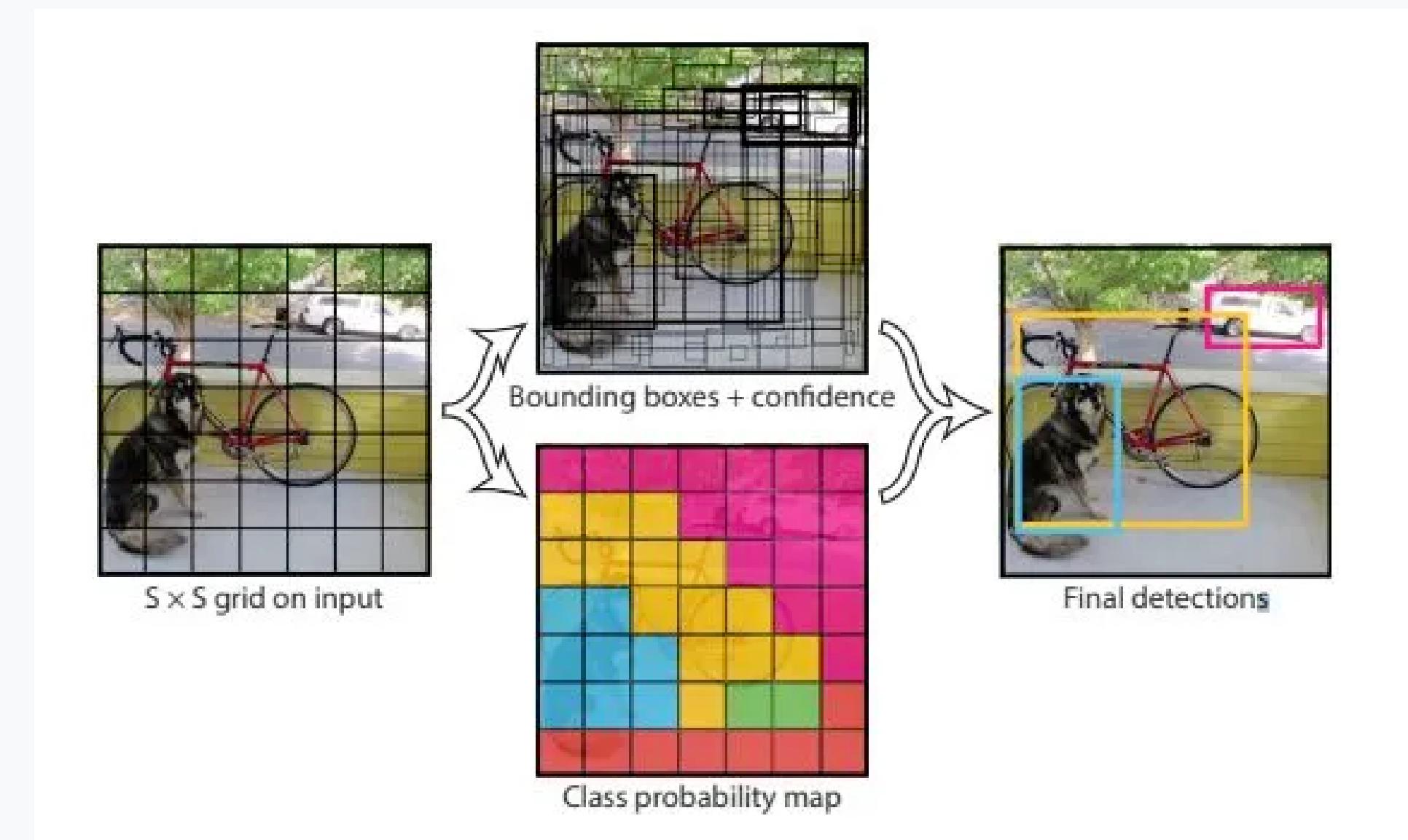
# Yolo - You Only Look Once

Yolo (You Only Look Once) 是一種物體檢測算法，它在單一前向傳播過程中能夠實現即時高效的目標檢測。

1. 實時性能
2. 全局上下文
3. 簡單而有效

缺點是對小物件的偵測效果較差，且目標定位不夠精確。

將圖像分為網格並在每個網格上預測目標的位置和類別。  
相較於傳統的檢測方法，不需要提取候選區域，而是直接在整個圖像上進行預測。



# Data Collecting & Augmentation

## 資料蒐集：

辨識的商品會以日常常見的零食飲料為主，同時選擇顏色清楚不複雜，且非透明包裝產品，以提升在模型辨識的效果，並且，和一般使用JSON檔案不同，YoloV8需要將標註檔案輸出為 \*.txt 檔。

## 資料擴增：

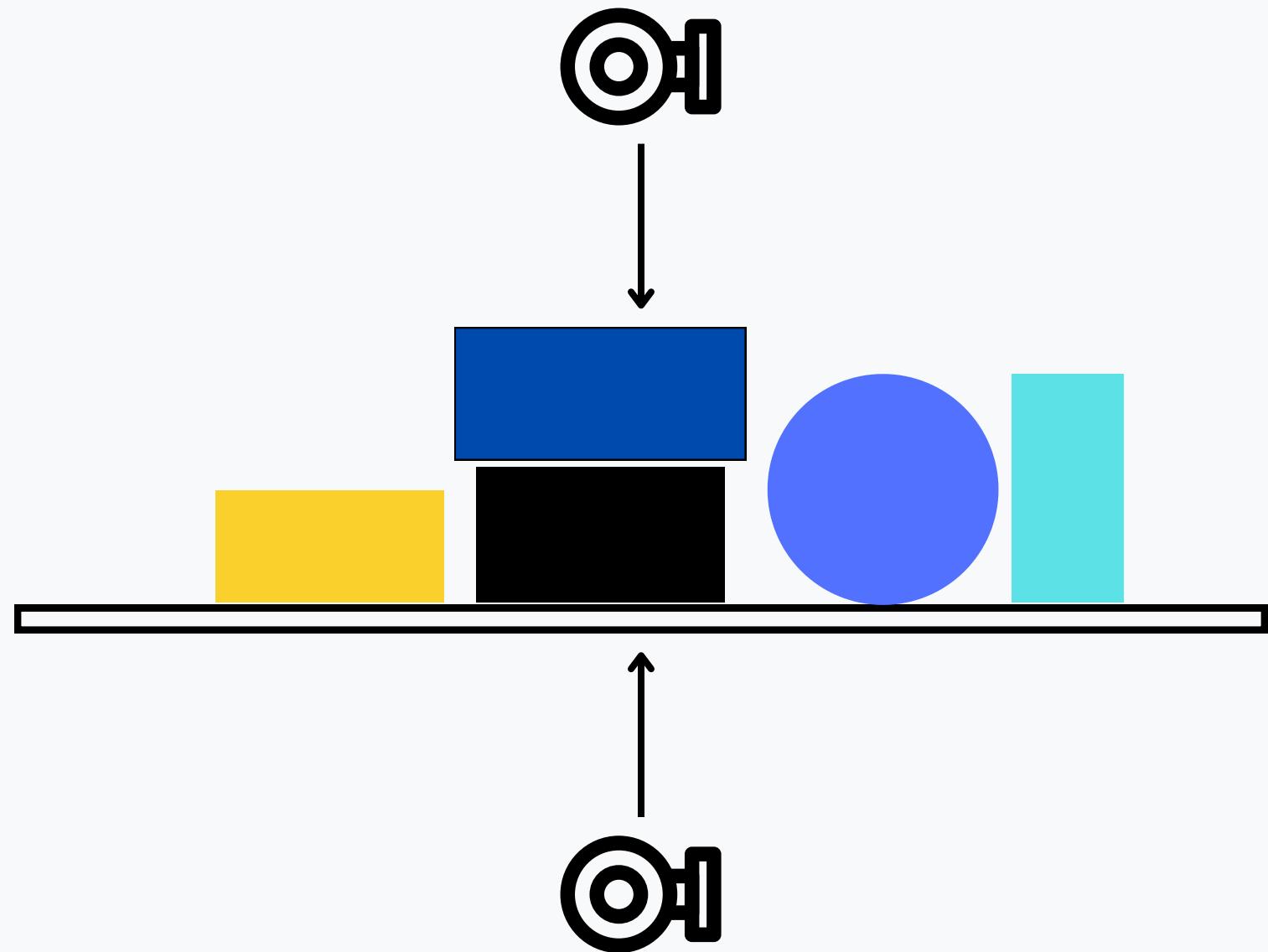
在實時辨識中，可能會因為擺放角度，光線等等原因，影響辨識效果，因此將原先圖片進行翻轉(水平或垂直)、旋轉、隨機亮度和對比度的調整、改變色調和加入高斯模糊，以提升在不同場景下的辨識表現。



# Avoid Items Stacked

## 物品重疊問題：

為避免物品重疊，導致辨識出錯，因此在物品擺放平台（透明）[上下同時使用鏡頭進行捕捉辨識](#)，若兩者辨識結果不一致，則提醒使用者重新放置。



右圖上方鏡頭因為黑色物品被重疊，因此不會辨識出黑色物品，而下方鏡頭則不會辨識出藍色物品

# Deployment

---

# What is Jetson Orin Nano

## NVIDIA Jetson:

專為嵌入式系統和邊緣運算，主板上包含一個小型GPU，使得在嵌入式設備上能夠進行深度學習模型，主要用於執行機器學習、深度學習和人工智慧工作負載。

Jetson Orin Nano使用操作系統是基於Ubuntu 22.04的Jetson Linux，將Ubuntu版本的App轉移至Orin Nano上，同時配置Pytorch和Cuda即可使用GPU進行加速。



# Conclusion

---

# Conclusion

## 1 提升臉部辨識效率：

若使用者超過一定數量，在進行臉部相似度比較會花費更多時間，導致判定效率降低，同時在臉部編碼儲存上也會造成負擔。後續可以採用臉部編碼分群以及向量資料庫進行改進。

透過分群分為  $K$  群，在實時辨識時，先跟各群平均向量做比較，找到最為接近的  $L$  個平均向量，並和該  $L$  群比較。

未改善判定次數為： $N$ ， $N$  為總使用者數量

改善後判定次數為： $K + \sum^L n_i$ ， $n_i$  為第  $i$  群中使用者數量

## 2 提升物件辨識準確率：

目前僅有物品正面圖片，若出現物品正反面差異較大，上下鏡頭辨識可能會出現問題，因此會增加物品種類及圖片，提升準確率。

# Thank You

