國立臺北教育大學
113 年度資訊科學系學生專題成果展
《專題報告》

作品名稱：CyCube：即時 3D 魔術方塊重建系統

指導老師：王人正

組　　員：李其祐

　　：陳奕

中華民國　113　年　12　月　20　日

國立臺北教育大學資訊科學系－學生專題成果展資料表

編號（由系辦填寫）	12	
作品名稱	CyCube：即時 3D 魔術方塊重建系統	
班級	資四甲	
指導老師	王人正	
組　　員		
姓名	手機	E-mail
李其祐	0961560509	lichyo003@gmail.com
陳奕	0982373839	eason.jshouse0725@gmail.com
指導老師簽章		

作品摘要：

魔術方塊廣受歡迎，但新手因高門檻常感挫折。現有教學方式（家教、團班、線上課程）各有其限制，難以兼顧成本與效率。我們開發了一套魔術方塊教學輔助系統，融合深度學習的手勢辨識與機器學習的顏色辨識技術，旨在降低學習門檻、提升教學效率。系統基於 SVM 模型進行顏色辨識，固定光線下準確率達 98.5%，並支持用戶自定義顏色以適應不同環境。手勢辨識部分利用 MediaPipe 和 LSTM 模型，達到 92%的精準度，並引入過濾機制以提升穩定性。此外，針對運行效能進行優化，包括減少畫面重疊以穩定幀率及縮短重建延遲，實現高達 106 FPS 的穩定表現。網路傳輸則透過後端優化與雙緩存技術，確保即時畫面更新與預測結果的同步性。結合 Git Flow 和 Docker 技術的開發流程，我們保障了系統的穩定性及功能快速迭代，為初學者提供了一個直觀、高效的學習環境。

中華民國　113　年　11　月　22　日

目　錄

一、前言

魔術方塊，這款經典的益智玩具深受大眾喜愛，不僅能鍛鍊空間思維，更能帶來無窮樂趣。在台灣，每年 World Cube Association（世界魔術方塊協會）舉辦多場公開賽，吸引眾多玩家參與。然而，新手入門卻面臨不小的挑戰：從魔術方塊的選購、公式記憶到解法思路，若無專業指導，學習過程漫長且容易受挫。

目前台灣的魔術方塊教學主要分為三種：家教、小班制團班和線上課程。家教雖能提供最即時的指導，但高昂的費用令許多人望而卻步。線上課程雖然價格相對親民，卻因網路環境、視訊品質等因素，影響學習效果。

二、研究目的

本研究旨在開發一套創新的魔術方塊線上教學系統，以解決目前台灣魔術方塊教學市場所面臨的高成本、低效率等問題。透過此系統，期望能提供給魔術方塊初學者一個更具互動性、更直觀的學習環境，降低學習門檻，提升學習成效。

三、文獻探討

(一)手勢辨識模型

動態的手勢辨識需要考慮手勢之間的時間序列，也就是在辨識時，除了基本的圖像擷取、圖像分割、特徵提取、手勢辨識等動作外，還需考慮到先前的動作。陳治宇在 2003 年提出了一種新型的人機介面[1]，利用影像辨識來取代滑鼠的功能，讓使用者不需依靠滑鼠也能操控電腦。文中使用隱馬可夫模型判斷各畫面的動作，並結合有限狀態機分析可能的行為。然而，文中提及在相似的動作上無法有效的區分，且對於影像處理的速度也僅有 5 幀，無法滿足如魔術方塊轉動手勢等需要高幀率的應用。

張文強結合了時序圖和雙流卷積網絡，透過兩個卷積網路分別對視訊的光流場和時間序列進行建模。一個網路專注於從光流場中提取當前的特徵，另一個網路則處理時間序列資訊。透過將光流場提取的特徵與時間序列網路的輸出融合，以判斷出目前動作[2]，不過雙流卷積網路仍存在如特徵提取不易、面對複雜場景時易被雜訊影響等缺點[3]。

在管維凱的研究中[4]比較了 RNN、LSTM 以及 GRU 等三種模型對於人體跌倒辨識的效果，以及對於最大最小正規化、相對位置正規化、線性插值加上相對位置正規化等不同的資料處理方式對於準確度的影響，在的結論中，比起 RNN 與 GRU，LSTM 對於人體節點的連續變化有著更佳的表現，且文中提到的正規化方式比較起原始模型皆有顯著的提升。最終我們結合張文強與管維凱的研究，先對擷取影像資料的關鍵特徵點(手部節點)，再利用 LSTM 對手部節點的位置建模。

1. 手部節點抓取

為了持續追蹤使用者的手部動作，我們參考了 MediaPipe 以及 Open Pose 等框架；OpenPose 有著較高的精確度，但計算量較大，需要花費較長的運算時間，而 MediaPipe 計算量較小，運算速度較快，能降低硬體負擔並提供更快的回應時間，在精準度及運算時長的衡量下，為了完善使用者的體驗，我們選擇 MediaPipe 這個套件進行開發，以達到低延遲的回饋。

2. 長短期記憶(Long-short term memory) [5]

Long-short term memory (LSTM), 是一種 Recurrent Neural Network (RNN)的變體，透過專門的記憶機制解決傳統的 RNN 在處理長時間序列時會遭遇的梯度爆炸或梯度消失的問題，在處理長時間序列的問題上通常有著比 RNN 更佳的表現。

基本的 LSTM 包含記憶細胞(Memory Cell)和三個閘門單元(Gate Units)，分別為輸入閘(Input Gate)、輸出閘(Output Gate)和遺忘閘 (Forget Gate)。

記憶細胞負責存儲訊息，包含細胞狀態 (Cell State)，即長期記憶，以及隱藏狀態 (Hidden State)，即當前時間步的短期記憶。其中細胞狀態保存了跨時間步的訊息，並依據輸入閘與遺忘閘的結果更新狀態，隱藏狀態則是細胞狀態經由激活函數處理過後的結果，反映了當前時間步的輸出，可作為當前 LSTM 單元的輸出或傳遞到下一時間步。輸入閘負責決定當下

的輸入訊息是否值得更新到記憶細胞成為長期記憶。輸出閘控制哪些訊息需要從細胞狀態中被提取，並輸出到模型的下一層或是下一個時間步，遺忘閘則負責決定記憶單元中哪些信息應該被遺忘或降低權重，並更新記憶細胞的內容。其中三個閘門皆為 Sigmoid function, 將輸出控制在[0,1]之間，以降低梯度爆炸或消失的風險。

(二)顏色辨識技術

在教學前，需要讓使用者可以快速掃瞄、上傳方塊狀態。在 [6] 這篇研究中，使用 OpenCV 套件對圖片做 Gaussian Blur, Canny Edge Detection, Contour Detection 抓到指定顏色的 Contour 並加以限制，刪除過大、過小的 Contour 。接著將 BGR 轉換成 HSV，透過把顏色和亮度分開，減少光線造成誤判的問題。最後透過座標相對位置，判斷魔術方塊狀態。

而在我們實作之後發現，此方式受限於以下幾點：

1. 預設 RGB 值域：

過大的 lower, upper bound 容易造成紅色、橘色的誤判。並且在此篇論文中似乎是使用同一顆方塊進行顏色判斷，然而在現今方塊多元的環境下，各種顏色已不再受限，因此會和預設的顏色值域差距過大，造成顏色判斷錯誤。

2. 光線環境問題：

在光源穩定的環境及單一方塊測試底下，我們可以透過此方式掃描方塊顏色；但是當光線或是方塊角度微調、使用者的手指遮蓋住方塊造成部分陰影，依舊會使之判斷精準度下降。

3. 計算量過大造成延遲：

因為使用 OpenCV 的 findContour 抓取指定方塊位置，但由於週遭環境有相近的顏色，此函式依舊會將之抓下來做一系列運算，即便最後大部分錯誤 Contours 會被刪除，但是依舊會造成執行上卡頓。因此在方塊顏色辨識的技術上依舊缺乏快速且支援多元環境的辨識功能。

(三)跨平台框架

框架名稱	優點	缺點	適合開發場景
Flutter	高效能、熱重載、強大的 UI 定制能力、豐富的 Widget 庫、Google 支持	生態系統正在快速發展，但相較 React Native 仍有差距	跨平台 UI 一致性要求高的應用
React Native	生態系統成熟，第三方庫豐富	性能較 Flutter 稍遜，但仍優於傳統 Hybrid App	利用既有 JS 技能的開發者

基於效能考量為優先目的，我們決定使用 Flutter 進行開發，即便其目前的 3D 資源未完善，但因為總體效能較好，以及可以和 Google 其他技術有更好的連接性，所以我們使用 Flutter 作為前端開發平台，寫出一個 3D 魔術方塊，以及架設一個線上教學系統 [7]。

(四)資料庫及串流系統

在應用程式開發中，選擇合適的資料庫和身份驗證系統對於提升性能和用戶體驗至關重要。根據 Patnaik et al. (2021) [12] 的研究，Firebase 提供的 Firestore 和身份驗證功能，使其成為開發現代應用的理想選擇 。首先，Firebase Firestore 作為一個 NoSQL 資料庫，允許開發者以靈活的方式存儲和查詢數據。Firestore 支持實時數據同步，代表數據變更可以即時反映給所有用戶，特別適合需要即時更新的應用場景 。其次，Firebase 的身份驗證功能簡化了用戶登錄過程，支持多種登錄方式，包括 Google 帳號和電子郵件密碼登錄。不僅可以讓開發者輕鬆管理用戶資料，還增強了應用的安全性。綜合來看，Firebase 的 Firestore 和身份驗證功能提供了強大的支持，使開發者能夠快速構建高效且安全的應用，滿足現代用戶的需求。

四、研究方式

(一)自“玩轉極限魔術方塊教育機構”蒐集老師意見回饋

1.「 線上教學魔術方塊時，學生常常因為網路延遲或鏡頭角度問題，導致我無法清楚看到他們的方塊狀態，教學進度很受影響。」

2.「 很難看清楚學生方塊的顏色，又會因為前置鏡頭的問題，難以分辨上下左右，每次都要花很多時間重複說明，影響人教學效率。」

3. 「 學生手指常常遮擋方塊，看不到顏色都只能用猜的去教學 」。

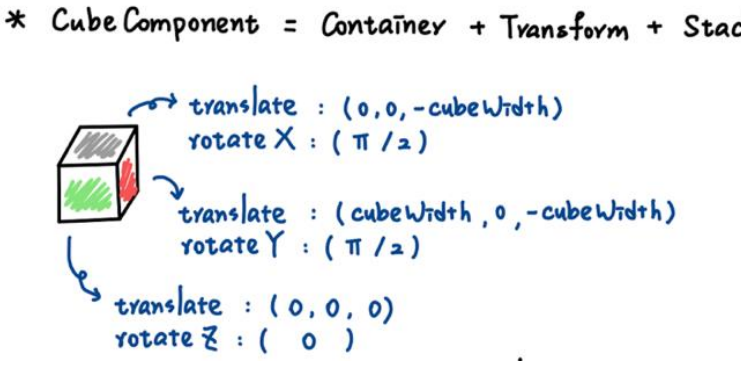
基於上述意見回饋，我們總結出來以下開發目標：

1. 開發 3D 方塊模型框架，供老師隨時查看任意角度的方塊狀態。
2. 可以透過手機鏡頭掃描，快速建立方塊狀態，並開發手勢辨識模型即時同步方塊狀態。

(二)前端模型開發

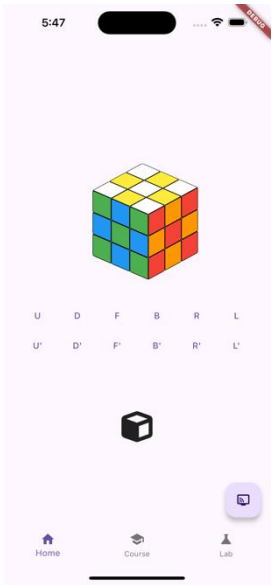
1. 利用 Flutter 的 Container, Transformer, Stack 製作 3D 方塊模型。

4-1 Single Cube Stack Arrange.



4-2 Cube Orientation Setup

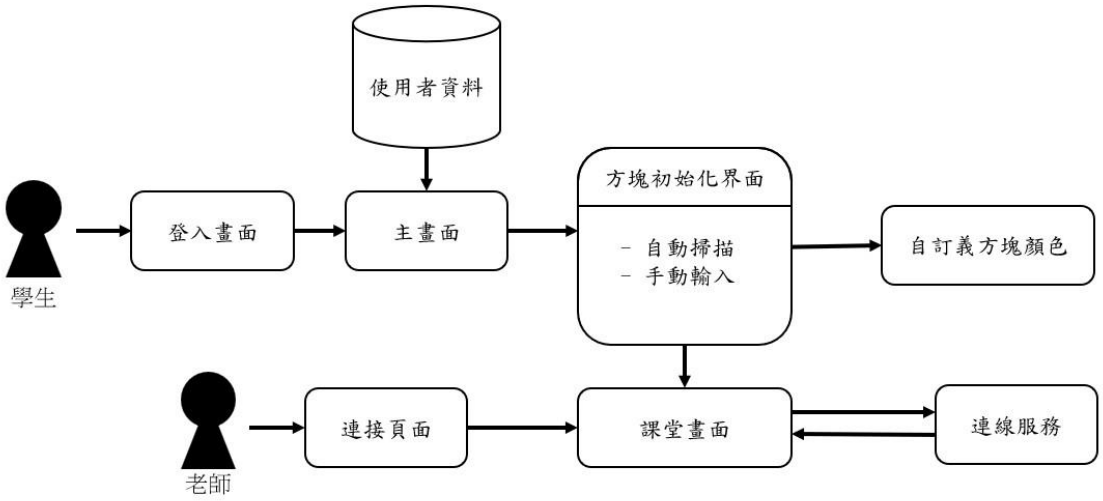
```
* Cube  = 2) CubeComponents + translate (x,y,z) + Stack
    for ( int x = -1 ; x <= -1 ; x++){
      for ( int x = -1 ; x <= -1 ; x++){
        for ( int x = -1 ; x <= -1 ; x++){
          cubeModels.add (
            CubeComponent (
              x :
              y :
              z :
              id : λ++
          ),
        );
    }
```



4-3 Compose Cubes.　　4-4 Cube

2. 串接 Firebase Auth 功能做使用者登入；

建立方塊初始化介面、自定義顏色介面、教學頁面，並串接 Flask App ，提供服務。

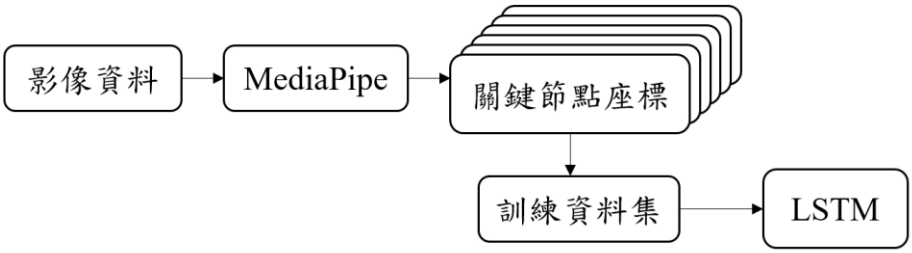


4-5 前端系統架構圖

(三)LSTM 模型訓練過程

此部分將針對提出的動態手勢模型做說明，並介紹資料預處理與模型訓練過程。首先將影像資料透過 MediaPipe 擷取出關鍵的節點座標，組合成具

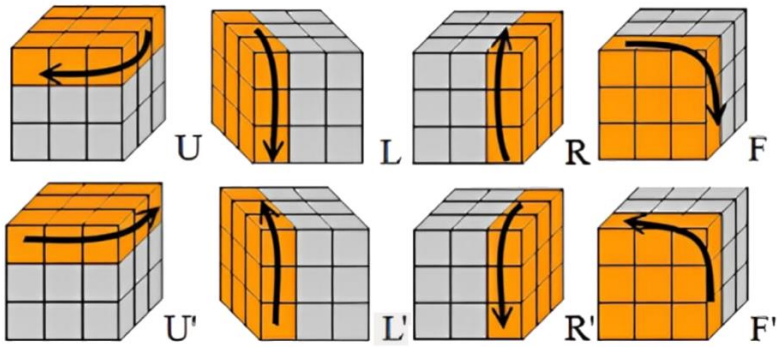
有時序性的關鍵節點座標資料，再透過最大最小正規化、相對位置正規化調整為訓練資料集，最後通過 LSTM 模型進行學習。



4-6 模型製作流程圖

1. 資料收集

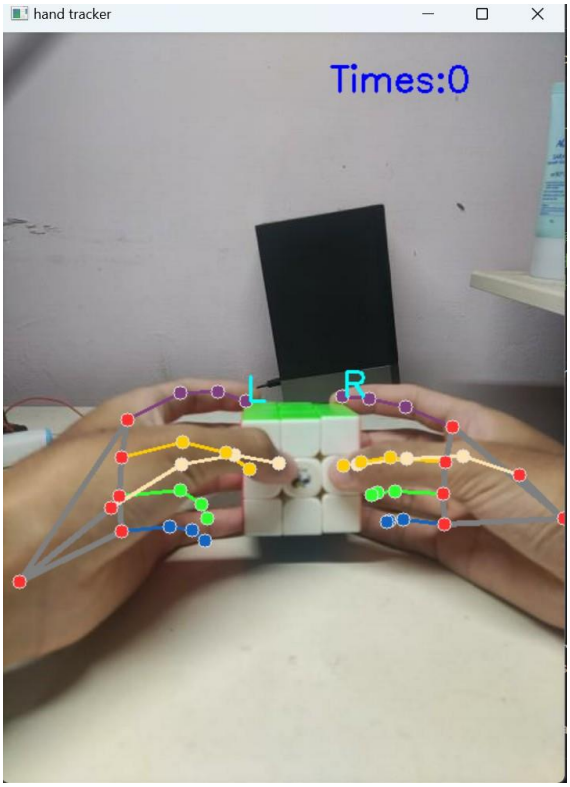
本次的資料皆為使用 3×3×3 的方形魔術方塊的轉動手型，以下的魔術方塊皆代表 3×3×3 的方形魔術方塊。本資料集包含常用的魔術方塊轉動手勢，包含 R, R', U, U', F, F', L, L', Wait 以上九種常用轉動方式。



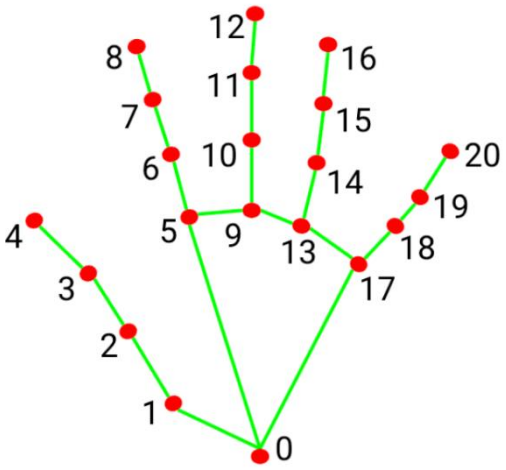
4-7 九種常用轉動方式 [8]

每種動作收集 1000 筆資料作為訓練集，200 筆作測試集，共 10800 筆。影片解析度為 680×480，皆以 30 幀錄製，且為了之後模型訓練方便，每段影片固定只擷取 21 個畫面。使用 MediaPipe Hand Landmarker，取得手部的 21 個節點位置(x, y)，如下圖，其中 1 到 4 號為大拇指，17 到 20 號為小拇指。由於鏡頭只能提供單一角度的畫面，使得 MediaPipe 在 z 座標的判斷上與 x,y 座標相比較不穩定，因此本次只採用 x, y 座標。

每筆座標資料包含 21 個畫面，每個畫面包含雙手共 42 個座標，座標以先左手再右手，joint 0 到 joint 20 的順序排列。



4-8 錄製轉動資料集



MediaPipe 手部節點示意圖 [9]

2. 資料預處理

為了降低模型訓練量並提高準確率，我們對訓練資料進行以下的預處理。

a. 位移座標

我們參考管維凱研究中的相對位置正規化[4]，將每筆資料的絕對座標都轉換成相對的座標，與管維凱做法不同的是，我們固定以第一幀的左手腕為原點，將包含第一個的每個時間步的座標都移動到新的相對座標上，並非對所有時間步的座標單獨調整，這樣能將訓練的重心放在手部的節點相對位置變化，從而減少因絕對位置差異所產生的特徵影響，也能將因轉動手勢而造成的整體位移納入模型訓練中。

b. 移除無用特徵與調整格式

在轉動的過程中，手掌節點的特徵對於動作的判斷並無明顯影響，且過多的無用特徵可能影響到模型的準確率，因此我們將代表手掌的節點(0、1、2、5、9、13、17)移除。

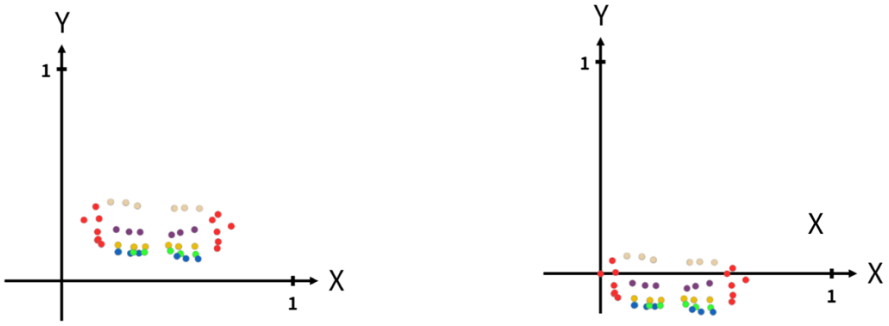
目前每個 time steps 的資料為一個 2×28 的矩陣 $M_{2\times28}$，其中 $M_{i,j}$ i=1 代表為 x 座標，i=2 代表為 y 座標，j 代表目前節點，1 到 14 為左手，15 到 28 為右手。由於 Keras 的 LSTM 輸入方式為一個三維的矩陣(N,T,F)，N 為資料數量，T 為時間步，F 為特徵，而我們一個時間步的資料包含 21 筆二維資料(x,y)，因此我們將每個 y 座標放在 x 座標後，將原先 2×28 的資料改為 1×56.1)。

$$M'_{1,i} = M_{k,[(i+1)/2]} \begin{cases} k=1, \text{ if } i \bmod 2 = 1 \\ k=2, \text{ otherwise} \end{cases} \qquad (4.1)$$

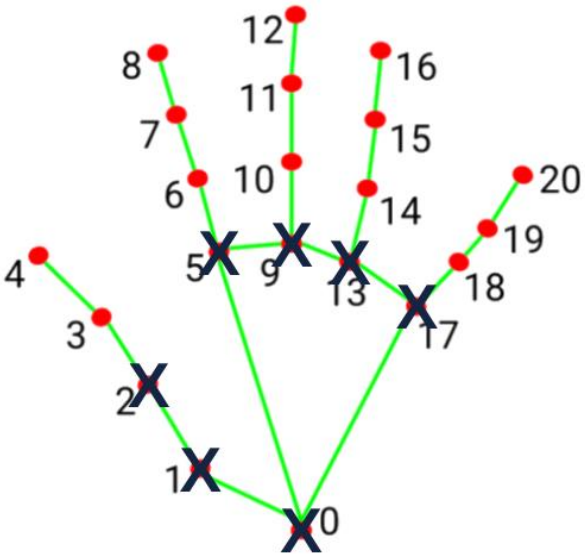
c. 最大最小正規化

為了提升模型的準確率以及加速模型的收斂速度，我們使用最大最小正規化，將輸入座標資料固定在 0 到 1 之間，如此也能解決因為距離或是個體差異導致畫面上手掌大小不同的問題。

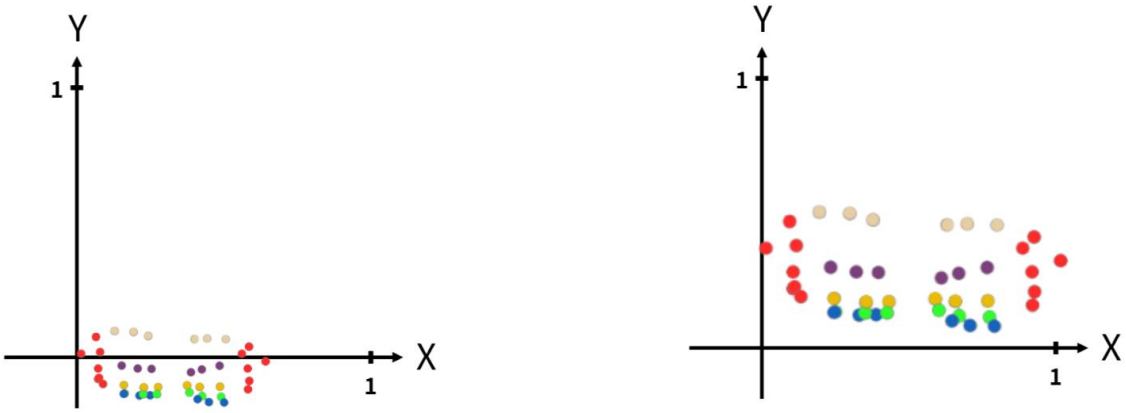
$$x' = \frac{x - \min(X)}{\max(X) - \min(X)} \qquad (4.2)$$



4-10 座標位移



4-11 移除無用特徵



4-12 最大最小正規化

3. 模型架構

我們設計了一個基於 LSTM 的神經網絡模型，用於將 9 種手勢分類。該模型由兩層組成：第一層為長短期記憶網絡 (LSTM)，單元數為 64，用以學習時間序列中的長期依賴性並帶有 L2 正則化(4.3)，正則化強度 λ 設為 0.1，以防止模型過擬合；第二層為全連接層，採用 softmax 激活函數(4.4)，將模型輸出轉化為多分類機率分佈。

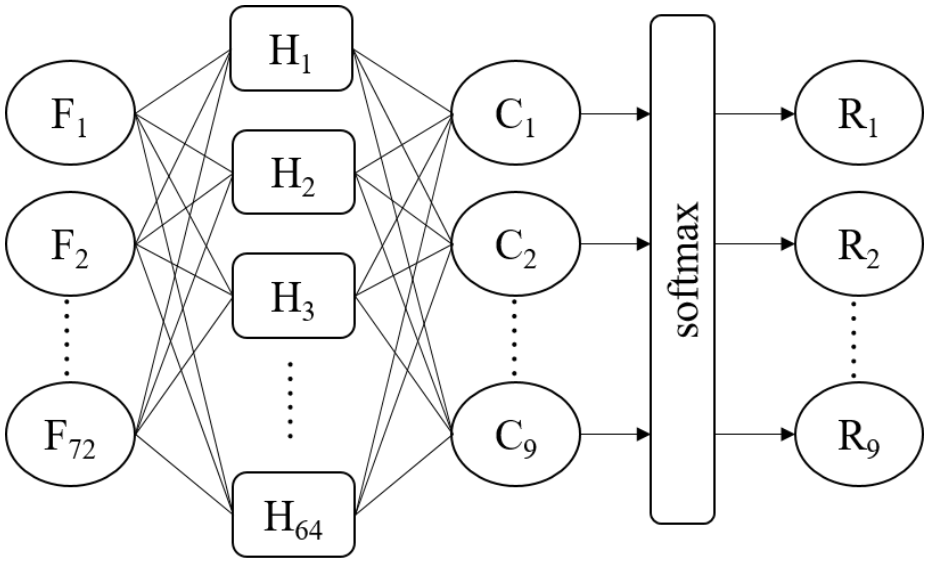
$$L_{total} = L_{original} + \lambda \sum w_i^2 \tag{4.3}$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K. \tag{4.4}$$

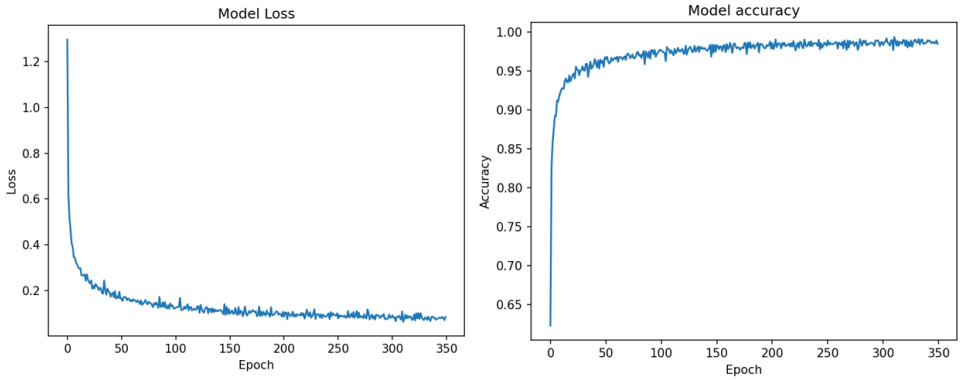
模型使用 Adam 優化器進行編譯，損失函數採用交叉熵(4.5)。

$$H(p, q) = -\sum p(x) \log(q(x)) \tag{4.5}$$

在訓練過程中，模型以 350 個訓練輪次和批次大小為 21 的方式對輸入數據進行學習，學習率設定為 0.001。下圖為模型架構示意圖，其中 F_i 為輸入特徵，H_i 為 LSTM 層單元，C_i 為全連接層輸出，經由 softmax 轉換為機率分布 R_i 即為模型輸出。



4-13 模型架構圖



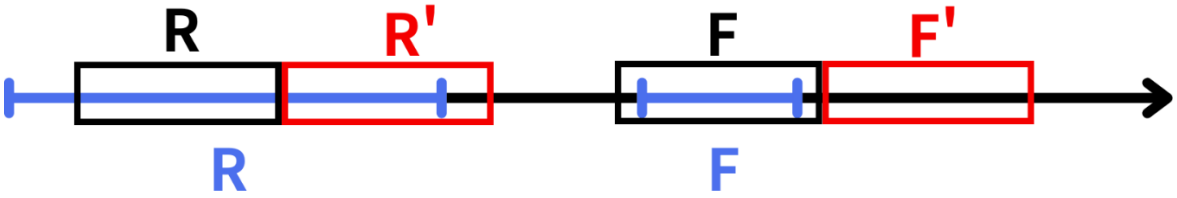
4-14 模型損失值(左)與準確率(右)訓練表現

(四)手勢預測系統

我們的手勢預測系統在單獨輸入轉動手勢至 LSTM 手勢辨識模型時，能達到約 92% 的準確率(此處準確率為測試資料集中所有的分類正確比例)。然而，在實際應用中，系統仍面臨兩大挑戰：如何準確辨識手勢的起始與結束點，以及避免產生不合理的預測輸出。為了解決這些問題，我們設計了一套改進的手勢預測系統。此手勢預測系統實時的抓取使用者的手部節點位置，並同時記錄先前 10 個畫面內的所有節點位置，一旦手掌姿勢的水平或垂直變動超過設定的閾值，則將紀錄的 10 個畫面加入預測資料，並開始記錄接下來的手勢節點位置，直到預測資料達到 21 個時間步，其中水平閾值為手掌最遠兩點距離的 0.09 倍，垂直閾值為手掌最遠兩點距離的 0.12 倍。由於模型每次預測需花費約 0.2 秒，即 1 秒只能處理 5 張圖片，不僅會對使用體驗造成影響，在預測上也可能發生關鍵的時間步遺失的問題。使用上述方法可以根據使用者動作對連續的動作做切割，且模型每秒最多只需計算 1 次，在能對各個動作做切割的同時也能保證影像能持續以 30 幀的速度處理。除此之外，對於預測系統的輸出我們做出了以下限制：

1. 阻擋相反動作：

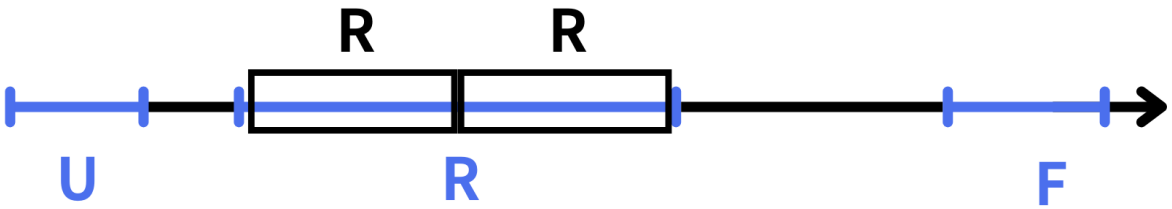
在每次轉動完成後需將雙手復位，在此過程中時常導致系統誤判成與原動作之相反動作，如:R 與 R’、L 與 L’，因此系統會將短時間內出現的相反動作視為無效輸出。



4-15 阻擋相反動作

2. 阻擋短時間的相同動作：

若連續手勢的斷點發生誤判，或使用者轉動的時間較長，導致一個動作跨越了兩次的判斷週期，則此系統會將連續兩次的動作視為單一動作，如下圖。若發生此情況，則第二次的預判結果會被捨棄。



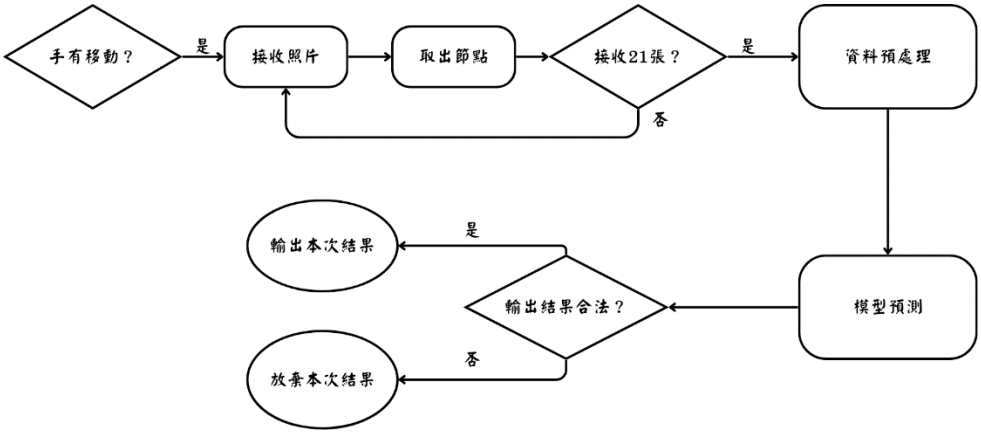
4-16 阻擋短時間的相同動作

3. 阻擋模糊不清的動作：

由於模型的最終輸出會通過 softmax 函數轉換為各手勢動作的機率值，為了降低誤判的風險，以及使用者在一次的預測內執行多項動作或做出辨識系統無法準確辨別之動作，此系統將機率值低於 0.7 的輸出視為無效動作。此預測系統在接收到無效動作時會忽略輸出結果，保證不合理的輸出結果不會影響使用者體驗。

最終系統的工作流程圖如下，若檢測到手指移動超過預設閾值，則收集 21 個畫面並提取手部節點組合成一筆預測資料，將其進行位移、移除無用特徵、調整格

式、正規化等預處操作後將其經由事先訓練好的模型進行運算，最終判斷結果是否為有效輸出，若為有效輸出則傳遞到 Firebase 進行後續處理，否則捨棄該次預測結果。



4-17 手勢辨識系統流程圖

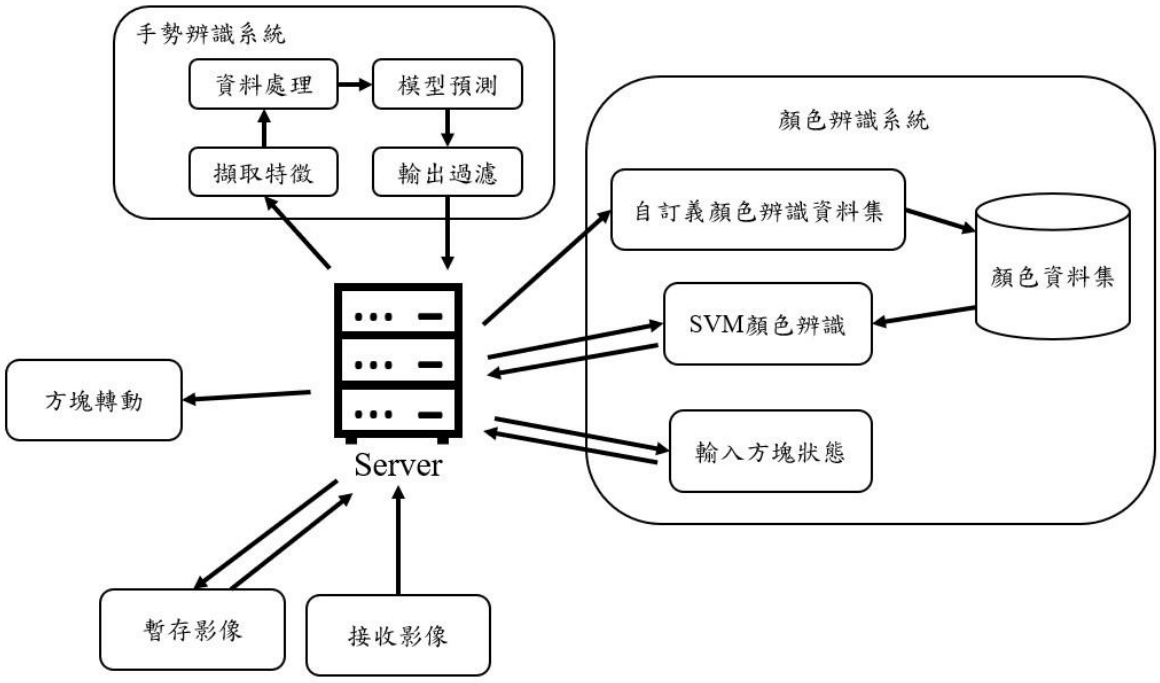
(五)顏色辨識系統

在先前文獻回顧段落中，我們針對 OpenCV 抓取指定顏色輪廓的方式進行實作及討論，然而在不同環境下造成精準度下滑的因素過多，所以後來並沒有繼續這條路。參考 Xiang-Yang Wang 使用 Support Vector Machine 對車輛的顏色做分類的研究 [10]，其在不同環境下依舊保持高精準度的結果，證明使用 SVM 在不同光線條件環境做顏色辨識是可行的，於是我們最後採用這種方式進行實作。為了使用 SVM 做顏色辨識，我們先搜集在各種環境下的六種顏色（白、黃、紅、橘、藍、綠）共 2160 筆 BGR 資料，作為訓練的資料集。我們從使用者的後置鏡頭擷取圖像，送至後端服務器進行處理並接收回傳圖片。並且透過雙緩存技術，實時的將處理後的影像顯示在螢幕上，同時也把同一面的九個預測結果回傳。最後，若是使用者使用方塊顏色較為特殊(i.e. 黃 -> 黑)，我們也提供自訂義顏色功能讓使用者透過鏡頭對著方塊做資料搜集，抓取 5 秒共 70 筆左右的資料加進或取代原資料集做訓練。

(六)Flask 處理各項服務

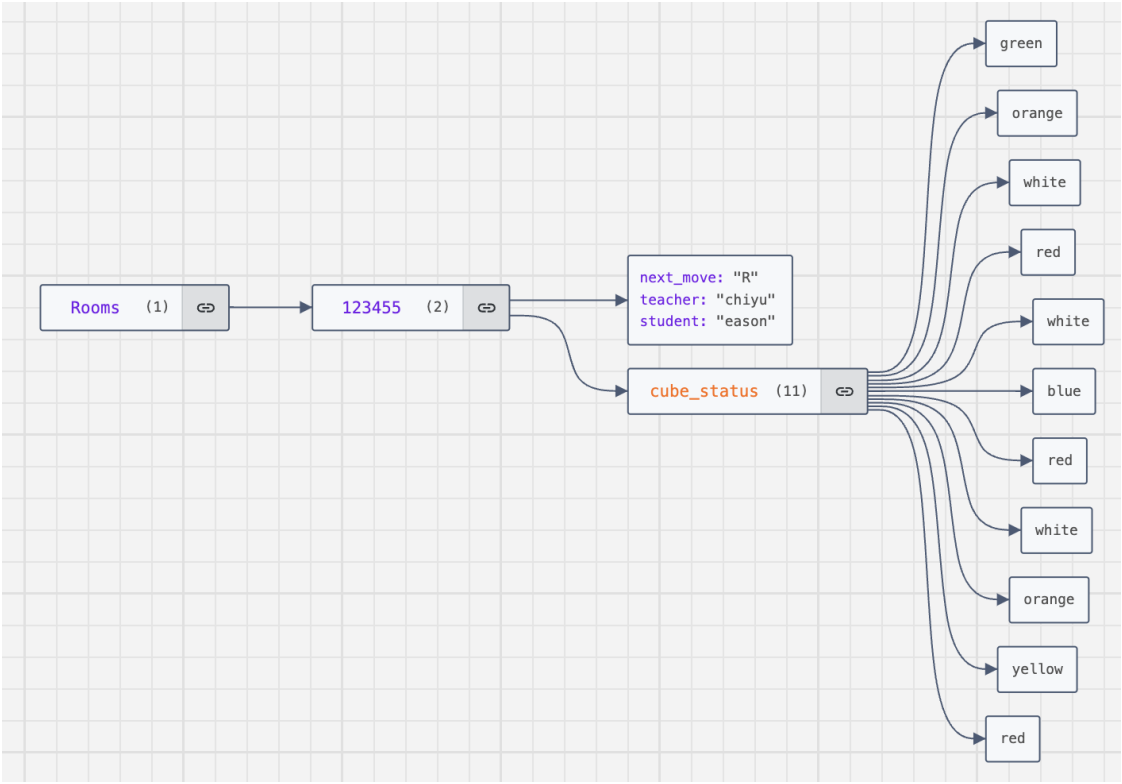
為了做到低延遲的服務，我們使用 Flask 作為 Web 框架，搭配 WebSocket-IO 套件做資料的串接，並且分成以下幾個事件：

1. save_image：使用者端鏡頭擷取圖像傳送過來，透過 base64Decode 成 Uint8List 型別資料並儲存為$user.jpg
2. receive_image：將處理完的$user.jpg 存成二進位檔加上 base64Encode 回傳客戶端。
3. initialize_cube_color：將$user.jpg 的圖片做顏色預測的處理，回傳一個預測陣列，例如：[“white”, “white”, “white”, “yellow”, “yellow”, “yellow”, “white”, “white”, “white”]表示由左至右，由下至上的九個顏色預測結果。
4. init_cube_dataset：從$user.jpg 抓取特定位置的 BGR 值並 Append 在 user_define_colors/$user.csv 內，作為顏色分類預設資料集。
5. clear_color_dataset：將 user_define_colors/$user.csv 資料清除
6. rotation：將連續 21 張$user.jpg 的使用者手部節點座標取出，並將[3, 4, 7, 8, 24, 25, 28, 29]點作為特徵進行轉動代號預測，並回傳以下預測結果[“R”, “R’”, “U”, “U’”, “L”, “L’”, “F”, “F’”,”wait”]



4-18 後端系統架構圖

(七)Firebase 資料結構及課程相關功能



4-19 Firebase Data Structure Example

名稱	資料型態	儲存資料	用途
rooms	collection	Room ID	加入教學區塊
cube_status	List<String>	54 個方塊顏色	建立方塊模型
next_move	String	轉動代號	監聽資料，更新雙方方塊狀態
teacher	String	教師	判斷是否可開啟課程
student	String	學生	判斷是否可開啟課程

首先透過顏色辨識系統讓學生上傳魔術方塊狀態至 Firebase，並存至 List<String> cube_status，並回傳隨機 6 位數作為 roomID，讓老師可以搜尋並加入。當老師加入後此

Channel 後會監聽 next_move 的變化，當學生轉動時將會更新此欄位狀態；而 Firebase 基於 WebSocket 的協定，可以達到低延遲、低耗能但高速且穩定的

(八)伺服器架設及部署

為了提升系統的可擴展性和部署的便利性，我們將後端服務透過 Docker 容器化，並運行在 Jetson Nano 上。這樣的架構不僅能夠簡化環境配置，還能確保服務在不同環境中的一致性。使用 Docker，我們能夠快速部署和更新後端服務，並且能夠輕鬆管理依賴項和版本控制。此外，Jetson Nano 的強大運算能力使得我們的系統能夠高效地處理手勢辨識和顏色辨識的計算需求，進一步提升了整體系統的性能和穩定性。

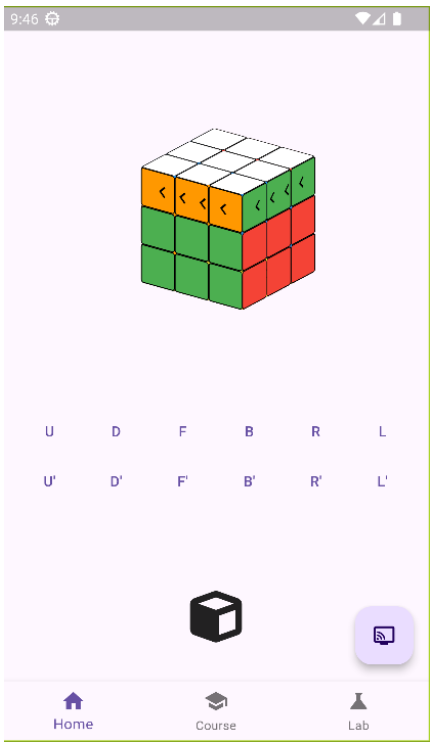
(九)開發流程

在本專案的開發過程中，我們採用了 Git Flow [11] 作為版本控制和協作開發的管理流程。Git Flow 是一種開發流程，能夠有效地組織和管理開發過程中的不同階段。除了使用 Git 用來記錄之外，我們也會把分支推到 Github 上面共同維護。而具體來說，我們使用了以下幾種主要分支：

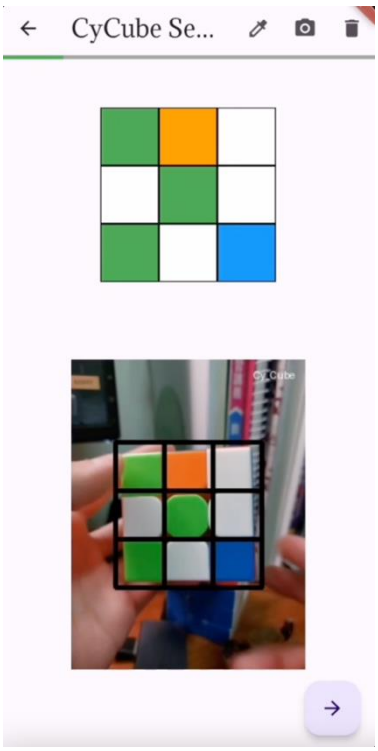
1. Dev Branch : 我們會從 main branch 拉下最新進度到 Dev branch 做開發，如果要開發的新功能比較複雜，就會在從這裡拉出去 feature 分支，例如：database, server 等等。做完一個特定功能做完測試後，就會將這個分支推到 Github 端。
2. Feature branch : 這裡會針對特定功能進行開發，開發完再回 Dev 分支，這樣就不會再測試的時候有太多的其他因素干擾，導致測試耗時太久。
3. Main branch : 這裏會放最穩定的版本，當要更新時就會看 Dev 分支有沒有推上來 Github 發 Pull Request ，只要有人發 PR 被 code review 過後就可以將這個開發分支合併回 Main ，並在 local 端拉下 main 分支開發。
4. HotFix branch : 當 Main branch 上面的服務遇到問題時，我們會在其之上開出 hotfix branch 進行快速修復，並在測試完成之後直接合併回 Main branch。

五、結果與討論

我們在研究中開發了一套專為魔術方塊教學設計的輔助系統，結合了深度學習的手勢辨識模型與機器學習的顏色辨識技術，解決了線上教學中高成本、低效率的問題，給予魔術方塊初學者一個更直觀且具互動性的學習環境。此系統具有顏色辨識功能，採用了基於 SVM 的模型對魔術方塊各面顏色進行分類。通過提取 BGR 顏色空間的特徵進行訓練，在固定光線條件下，顏色辨識準確率達到了 98.5%。對於不同光照環境，我們設計了用戶自定義顏色功能，允許用戶調整與更新模型，並且在一分鐘以內即可建立模型。



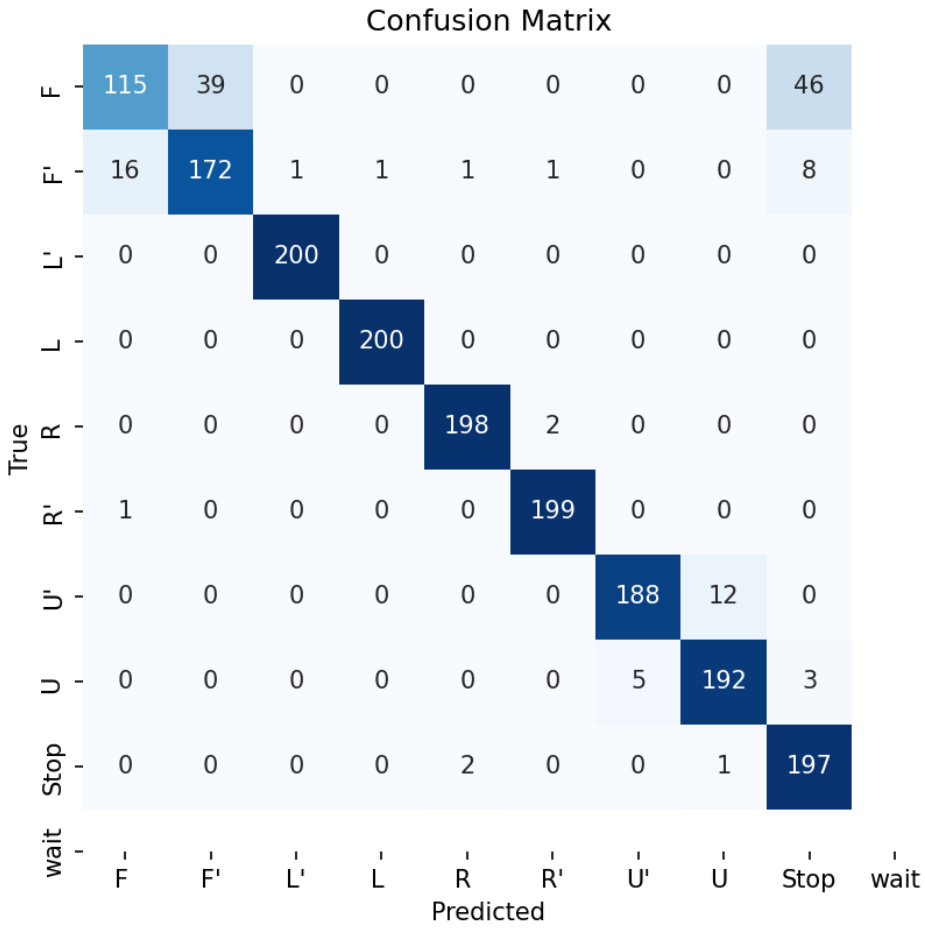
5-1 教學系統畫面



5-2 SVM 顏色辨識結果

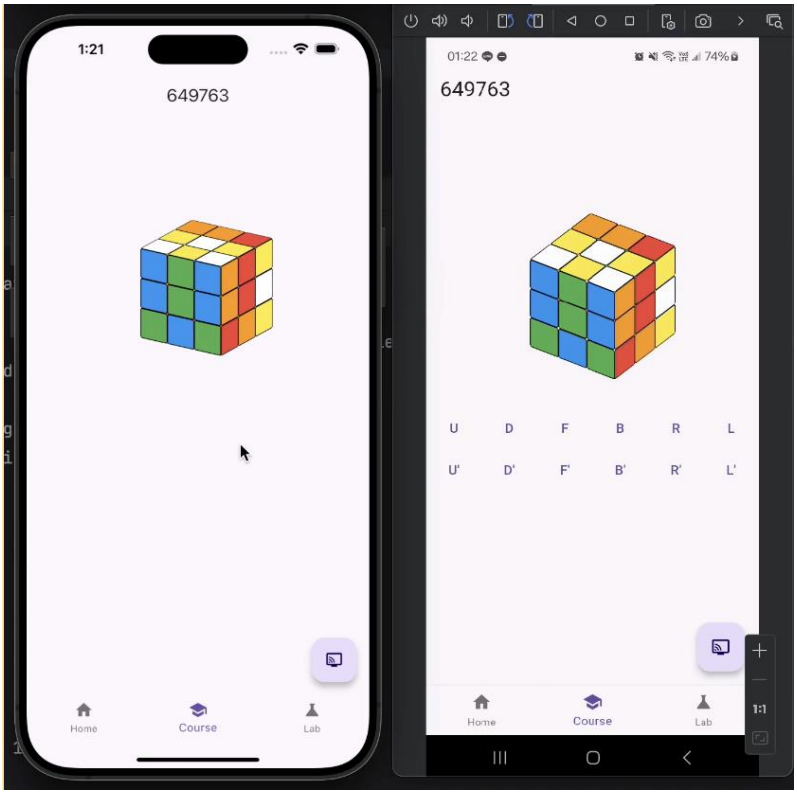
本系統整合了連續手部動作辨識功能，利用 MediaPipe 捕捉雙手的 42 個關鍵節點，將其轉化為時間序列特徵，並應用 LSTM 模型進行動作辨識，實現對 9 種魔術方塊手勢的分類，達到約 92%的準確率，我們通過混淆矩陣對模型的分類結果可以發現，相比於其他動作，該模型對於 F、F'較不敏感，但在其他動作上，如 L、R 等節點位置變動大、不易受到噪音影響的動作上有著極佳的表現，若忽略 F 與 F'則該模型可以達到 98%的

準確率。此外，在手勢辨識系統中還導入了額外的過濾機制，排除可能會影響使用者體驗的錯誤輸出，進一步提高系統在實際應用時的穩定性。



5-3 手勢辨識模型混淆矩陣

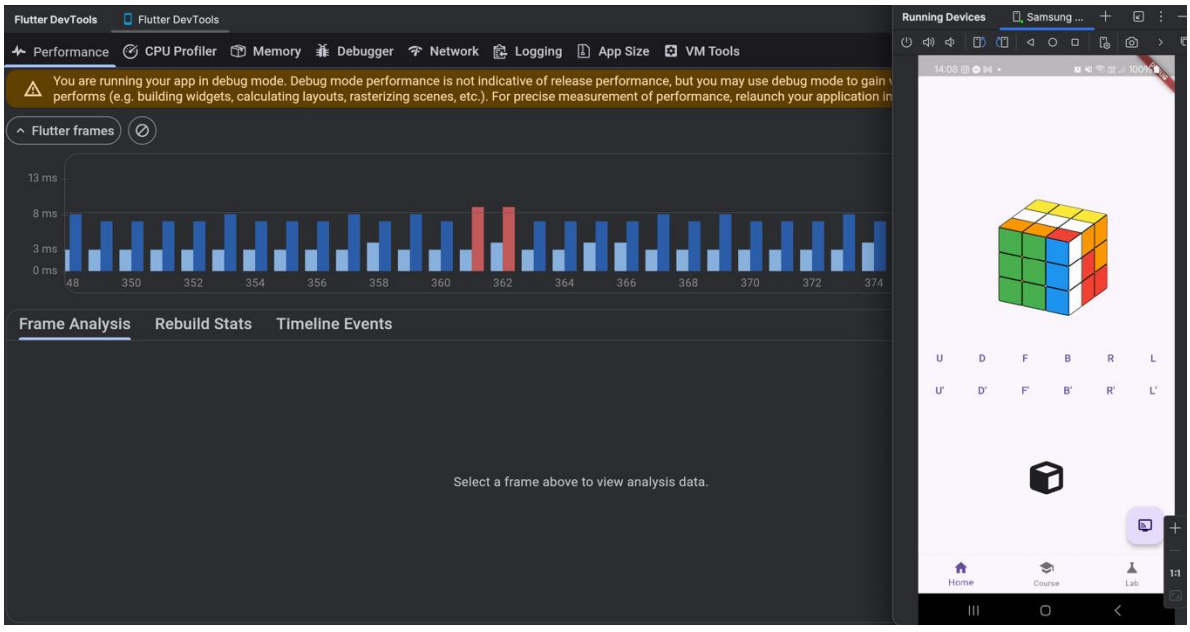
並且藉由 Firebase FireStore 的 監聽功能，我們針對不同裝置間同步方塊的功能，將延遲降低至 1114 millisecond，也就是說當學生觸發轉動事件，到老師端同步方塊，只需要耗時 1 秒左右的時間，達到近乎實時更新的目標。



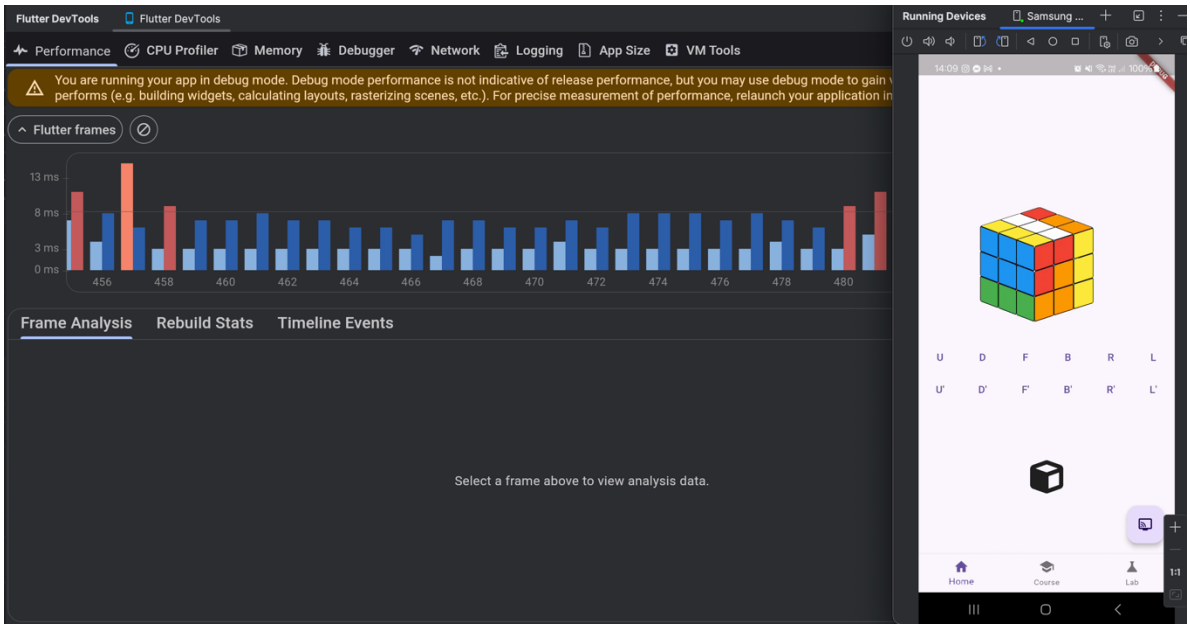
5-4 Firebase 同步不同裝置間方塊狀態

接著是關於系統穩定性的討論：

(一)主程式效能



5-5 轉動效能



5-6 翻轉（重建）效能

1. 幀率穩定：

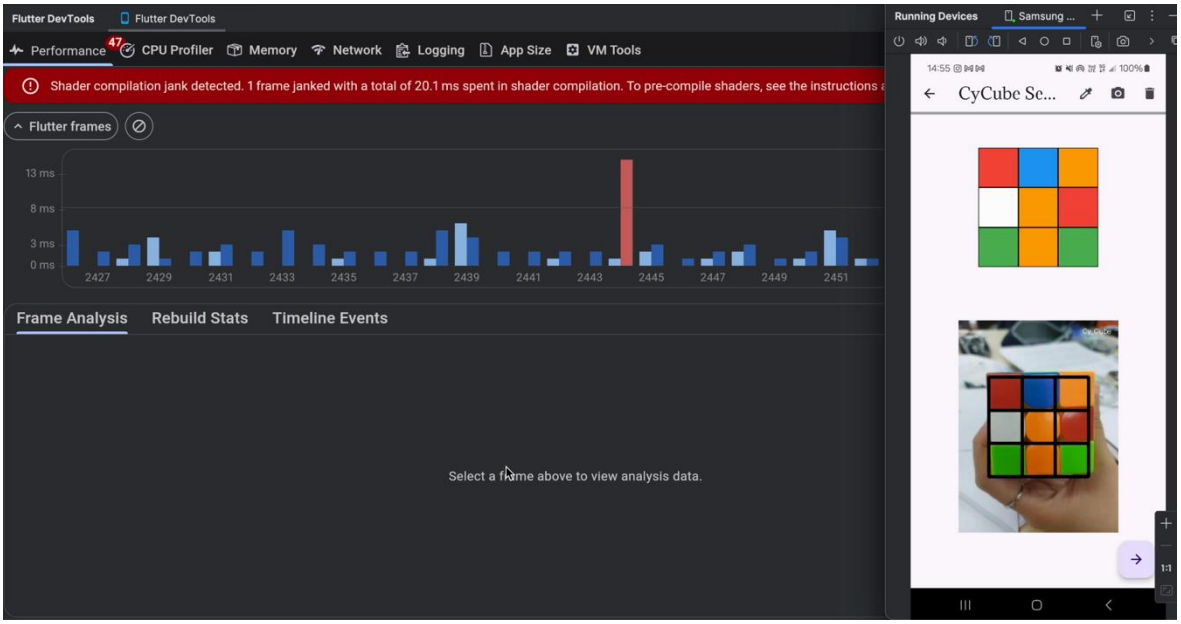
我們透過減少大量重疊畫面，縮小使用物件，讓大部分的幀渲染時間都保持在較穩定的水平，表示應用程式在大部分時間內都能夠達到流暢的畫面更新率。

2. 重建效能：

在翻轉時只有少數時候會因為重建需要將 27 顆小方塊重新排序和渲染畫面，所以在重建當下會有 18 ms 以上的延遲，甚至有掉幀的情形。但是因為前面提到的減少物件及縮減畫面重疊部分，因此就算在重建時，程式依舊可以穩定保持 106 FPS 以及至多 13ms 的延遲。

(二)網路效能

我們將後置鏡頭畫面傳送至後端服務器處理後，再將預測結果和相片一起回傳，而透過雙緩存的技術，我們得以流暢得顯示實時畫面及準確的預測結果。



5-7 顏色辨識系統效能

(三)後端服務穩定

得利於 Git, Git Flow 等工具和開發流程的幫助，我們可以在開發過程中藉由 Pull Request 和 Code Review 的機制讓系統始終在穩定的狀態。並且在系統要將新功能上線時，透過 Docker 開啟新的 Container 運行，一旦系統故障時，我們也可以即時開出先前穩定版本的 image 繼續運行服務。

六、參考文獻

[1] C. Zhiyu, "Virtual Mouse: Vision-Based Gesture Recognition," M.S. thesis, Dept. of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, 2003.

[2] Z. Wenqiang, "Human action recognition combining temporal dynamic graphs and dual-stream convolutional networks," Laser & Optoelectronics Progress, vol. 58, no. 2, pp. 0210007-1–0210007-9, Jan. 2021.

[3] H. Jinhao,” Action Recognition of RGB Videos Based on Deep Learning”, M.S. thesis, Dept. of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan, 2020.

[4] K. Weikai, "Research on Fall Detection Based on OpenPose Skeleton and LSTM/GRU Models," M.S. thesis, Dept. of Information and Communication, Chaoyang University of Technology, Taichung, Taiwan, 2020.

[5] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[6] J. Šuklje, *Aplikacija računalniškega vida za reševanje Rubikove kocke v realnem času*, Diplomsko delo, University of Ljubljana, 2023. [Online; accessed 13-Aug-2024].

[7] W. Wu, "React Native vs Flutter, Cross-Platform Mobile Application Frameworks," Bachelor’s thesis, Information Technology, Metropolia University of Applied Sciences, Helsinki, Finland, Mar. 1, 2018.

[8] 小螞蟻的學習筆記, https://ant2016.blogspot.com/2016/03/33.html, [Accessed: Dec. 1, 2024].

[9] Google, “Hand landmarks detection guide”, Google AI for Developers, https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker, 11/2024

[10] X.-Y. Wang, T. Wang, and J. Bu, "Color Image Segmentation Using Pixel Wise Support Vector Machine Classification," *School of Computer and Information Technology, Liaoning Normal University*, Dalian, China, and *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications*, Beijing, China, 2016.一·

[11] S. Chacon, 為你自己學 Git. 台北: 旗標出版, 2014.

[12] Patnaik, R., Pradhan, R., Rath, S., Mishra, C., & Mohanty, L. (2021). Study on Google Firebase for Real-Time Web Messaging. In D. Mishra et al. (Eds.), *Intelligent and Cloud Computing* (pp. 461-469). Springer Nature Singapore Pte Ltd.