ES\_Final\_Project—手勢控制音樂播放器

B07901166 程騰緯

B08901059 吳孟宸

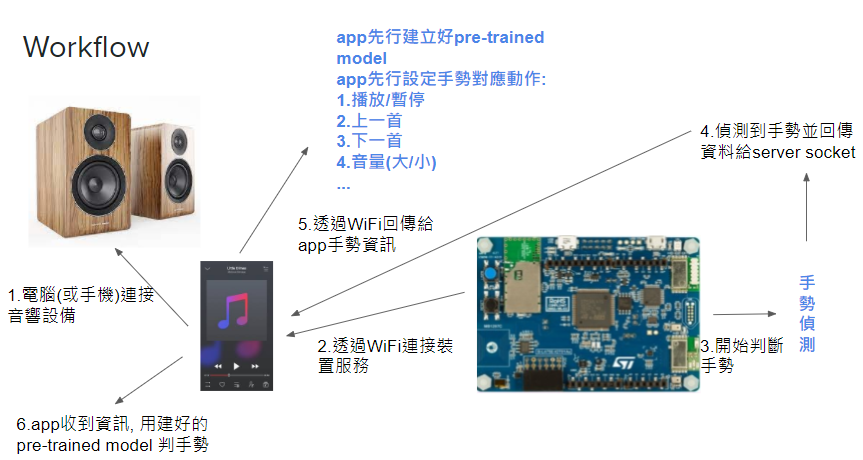
B08901040 張嘉恆

一、動機：

* 在這個科技進步的時代，使用藍芽耳機聽音樂對我們而言已經是一種常態。不過當我們在家時，偶爾還是會想用家裡的音響來聽音樂吧？
* 然而，有時候我們的「懶癌」發作，讓我們不太想動身去操作音響。所以這時候，STM32就派上用場了！有了STM32，即使在遠處，我們依然可以利用手勢操作來播放、暫停音樂；或者是轉上、下首；以及調整音量等等，讓在家聽音樂變得更方便。

二、基礎架構：

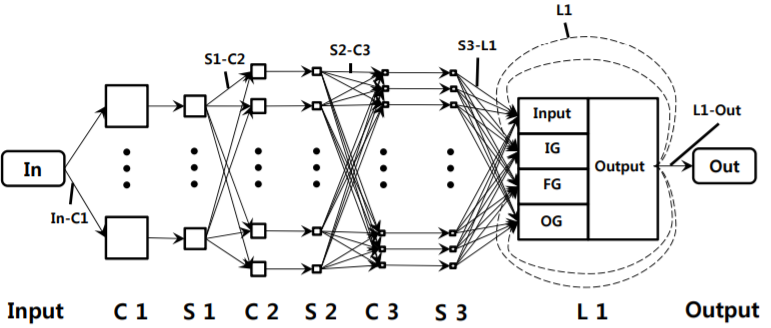
1. server-end (python) :
   1. 利用pygame.mixer建立簡易的音樂播放器
   2. 利用tkinter建立UI介面
   3. 利用WIFI和client端建立連結，進行資料傳輸
   4. 參照pre-trained model，判讀由client端傳過來的資料是指向哪一種手勢，並由此得出要進行哪種動作
2. client-end (STM32) :
   1. 用STM32接收加速度資料
   2. 當STM32接收到的data值 > 設定的threshold => 發現有手勢進行中，待手勢結束後將手勢過程中的加速度資料回傳給server
3. workflow :

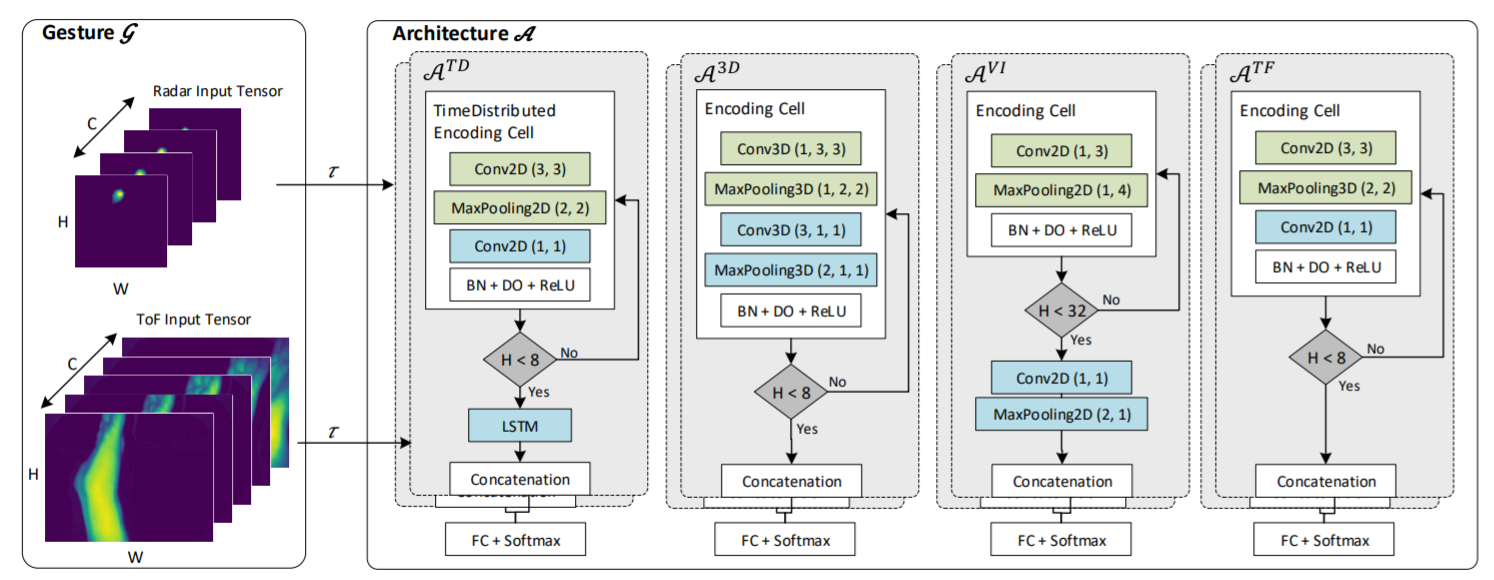


三、文獻探討 (paper survey)：SVM / NN models

* 實作重點 : 將input的data(三維加速度資料)轉換成1個整數(判別出的手勢)的output
* **尋找適合手勢判斷的資料種類及演算法:**
  1. 資料種類 : 三維加速度、三維角速度、影像...
  2. 演算法：簡單條件判斷、NN、SVM、DTW…
* 初步結論：使用加速度與角加速度，多種神經網路判斷[1]、使用加速度與DTW(dynamic time warping)判斷[2]、使用加速度和SVM(support vector machine)判斷[3]，三篇文獻皆在10種左右的手勢有90%以上的平均成功率，故我們認為三維加速度是充分的判斷資料。

1. SVM :
2. 以實作[3]的FDSVM(Frame-based Descriptor and multi-class SVM)方法為目標。
3. 前處理：將一筆手勢資料切成N個segments，每兩個segments為一個frame，共N-1個frames。對每個frame計算五個特徵值，{μ,ε,δ,σ,γ}。μ是DFT後的DC component，ε是DFT除DC外其他component的energy，δ是DFT後頻域訊號的information entropy，σ是原時域加速度資料的標準差，γ是兩維時域加速度資料的correlation。最終交給SVM的資料是一個frame有{[前4個特徵\*3維=12筆]+[3維任取2維的correlation=3筆]=15筆}資料，frame的數量參考[3]取9個，一筆手勢資料共135個資料點。
4. SVM model：經過前處理後的手勢資料以one to one的方式建立multi-class分類器。one to one: 對任意兩種手勢皆找到可當作區分標準的support vector，被所有二元分類器判定到最多次的種類為判定結果。
5. NN models :
   1. Find some CNN + RNN(LSTM) models for gesture detection for ( accelerometer and) **streaming video** ( [4], [5] )
   2. **Streaming video :** 首先要在每一個frame取出的照片之間提取出features (特徵) **(CNN part of the model)**，然後需要在這些取出的、帶有時間順序性 (sequential) 的 features data 中找出其對應的手勢 **(RNN、LSTM part of the model)**
   3. 找到幾篇paper的結果，顯示這樣的model architecture在手勢數量變多 (20 甚至超過 30) 的時候仍能夠有很高的accuracy
   4. 初步想法:建立RNN architecture, input accelerometer data (3-D) to train, output = a value (label of the gesture type)

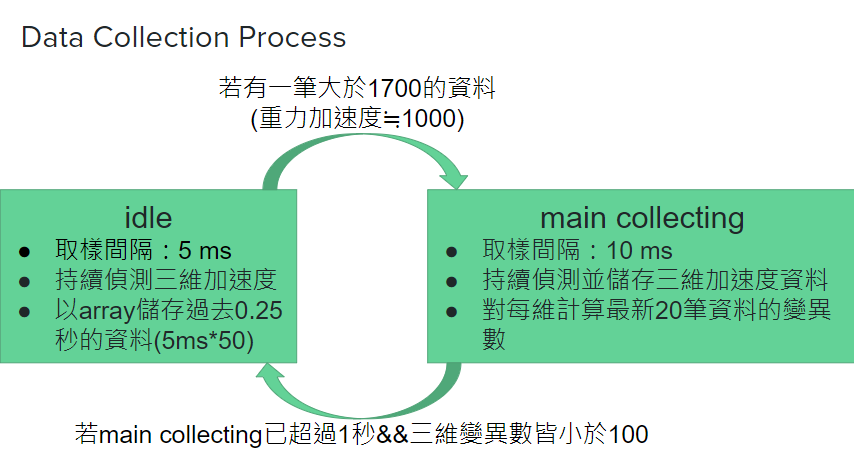




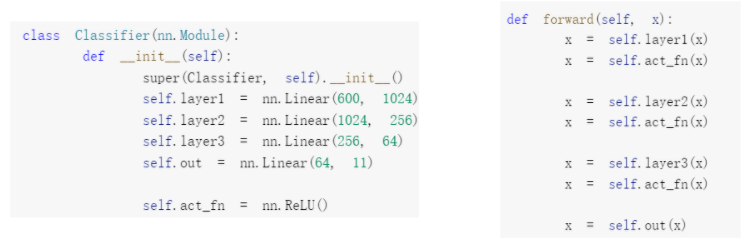
( model architectures in the ref. paper)

四、實作：

1. data collection (STM32):
   1. use **BSP library**
   2. 初期 : 揮兩下 — 第一下(相當於「開始」手勢)會讓STM32開始蒐集三維加速度資料(也就是第二下)
   3. 問題 : 揮兩下的動作並不實用 / 在第一下和第二下手勢之間的等待時間是固定的，可能會造成收集資料的問題
   4. 改善 : 利用 **threshold / idle 偵測手勢的開始 / 結束**，完成**非固定frame的資料蒐集** (詳見圖一)
2. data communication :
   1. use **wifi connection**
   2. 傳輸的資料 : **600筆 — 200個時間點的三維加速度資料**，取超過**threshold前的50筆(間隔:5ms)**和**手勢開始到結束的資料(間隔:10ms)**，若偵測到結束前未滿200點則補滿(取偵測的最後一筆(x, y, z)加速度分別補滿三維)
3. models(1) : SVM
   1. 前處理：運用**CMSIS-DSP的Real FFT函數**得到DFT再計算5種特徵
   2. 運用python的sklearn.svm.SVC得出每個二元分類器的support vector
   3. 計畫使用**CMSIS-DSP的SVM library**，以收集到的support vector建立跟python端一樣的分類器，可在STM32就完成判斷，僅需傳輸代表手勢種類的一個整數。
   4. problem (with only 5 types of gestures) : 各特徵的數值落差大，無法完整呈現手勢特性，實際判斷準確率低。嘗試改用未處理的加速度資料，成功率有提高但仍不足，為了支援更多種手勢，尋找其他成功率更高的辨識方法 ( NN )
4. models(2) : NN
   1. 運用pytorch.NN
   2. 先行建立簡單的一維傳統 NN 架構測試，發現效果非常好 (6種手勢時，train acc > 0.99，hands-on test的結果也完全正確)，決定延用架構。
   3. 架構: input layer/ 4x hidden layer/ output layer，input size = 600，output size = types of gestures (詳見圖二)
5. python (wifi server, music player) :
   1. 運用pygame.mixer / tkinter
   2. wifi connection : server socket (建好，等待STM32連線)
   3. pre-trained model: import pytorch.NN，建好model再load進來已經訓練過之model參數
   4. **use multi-threading : opened server socket and GUI(tkinter) at the same time**



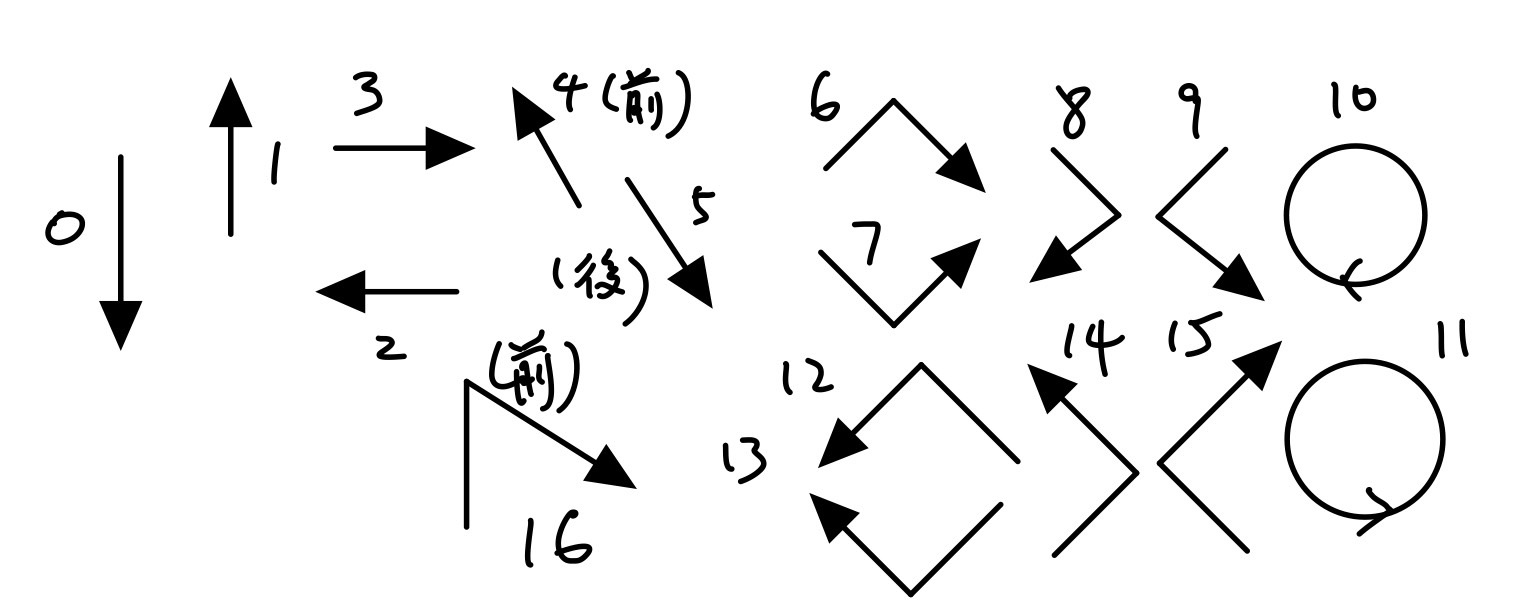
(圖一)



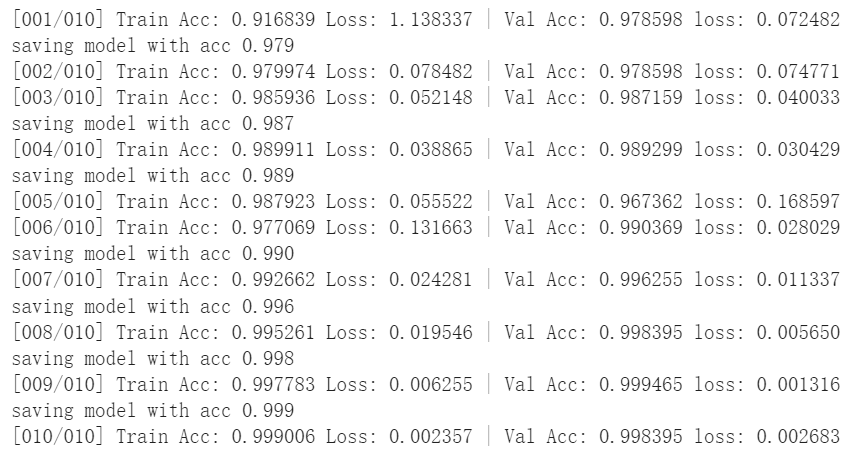
(圖二)

五、結果，及問題與未來展望：

1. 手勢 : 11種(demo) -> 17種(1/18 add) (圖三)
2. 結果:
   1. demo : **11 gestures** - **>99% training acc**，hands-on testing : **108/110 = 98.2%** (每一種手勢測試10遍)
   2. new implementation : **17 gestures** - still **>99% training acc** (圖四)，hands-on testing : **164/170 = 96.5%** (每一種手勢測試10遍)
   3. but the above is done with **fixed position of hand (that holding the STM32)**
3. 問題與未來展望:
   1. Data pre-processing: **bandpass filter** (消除DC的 offset ((重力加速度)) 和一些高頻的雜訊) (can use **CMSIS-DSP’s function**)
   2. 承上c.，我們的 train data 和 test data (包含 hands-on test) 都是在人上半身為直立且手平拿STM32開發板時得到的，但我們注意到人不是直立時且手沒有刻意保持平放時，手勢偵測的正確性會顯著的下降(大約80%，且像是有向左/右之手勢錯誤率會大幅上升)，所以**加入其他狀況時候的手勢資料來進行 model pre-training**，也是未來進一步實作可以考慮的方向。
   3. pygame.mixer只有包含非常簡單的library to access the music file (.mp3) (像是許多音樂播放器/耳機都會有用到的equalizer裡面就無法使用)，未來可以找尋功能更完整的library或是是否有現成音樂播放軟體的python API可使用 (這樣也可以有更好的UI)
   4. Duration of the gesture: for some gesture, the duration of them might too long : **use longer / flexible window**
   5. For **power efficiency**: 找到方法讓STM32不要一直收集input然後判斷(**不要讓他一直polling**)，可以讓accelerometer inputs 超過一個閥值 (second threshold ?)再將其打開開始蒐集data
   6. If **more gestures (20種以上) are required**，**only linear NN model may not enough - implement of RNN (LSTM) model**



(圖三)



(圖四)

五、文獻資料 (reference)：

1. Recognition of Hand Gesture Sequences by Accelerometers and Gyroscopes [1]
2. uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications [2]
3. Gesture Recognition with a 3-D Accelerometer [3]
4. Dynamic Hand Gesture Recognition for Wearable Devices with Low Complexity Recurrent Neural Networks [4]
5. Sensor Fusion Neural Networks for Gesture Recognition on Low-power Edge Devices [5]

六、實際展示 (demo)：

1. <https://www.youtube.com/watch?v=OxUirHtwkVc>
2. <https://www.youtube.com/watch?v=YXieVkSTB38>