國立臺灣科技大學電機工程系

109學年度第1學期實務專題

專題成果報告

手寫字跡矯正墊板

組　　別：　D07-1091

組　　員：　姓名：劉錕笙　　學號：B10607118

姓名：陳儀銘　　學號：B10607133

姓名：施丞祐　　學號：B10607121

指導老師：  (簽名)

中華民國 109 年 12 月 25 日

目錄

第一章　緒論 3
1.1　研究背景 3
1.2　研究動機 3
1.3　文獻回顧 3
1.4　專題貢獻 5
第二章　理論背景 6
硬體設計 7
(一)　概述 7
(二)　內容 7
手機應用程式 11
(一)　概述 11
(二)　流程 12
(三)　演算法介紹 13
深度學習模型 19
(一)　概述 19
(二)　流程 19
(三)　模型詳細內容 21
第三章　實驗數據 24
第四章　建議與結論 26
第五章　未來發展方向 26
第六章　時間進度表(甘特圖 Gantt Chart) 27
第七章　參考資料 28
第八章　工作分配 29
第九章　經費表 29

第一章　緒論

1.1 研究背景

看字如看人，好的字跡能給別人好的第一印象，也能使自己心情愉悅。然而現代人字跡卻普遍潦草。這個問題可能要歸咎在便利的通訊設備使現代人少了很多動筆的機會。

「練字」是每個人都會遇到的問題，然而在練習的過程往往只能靠不斷的臨摹字帖來慢慢的學習。對於剛開始練習的人來說，不論怎麼寫都達不到「漂亮」的標準，甚至連「工整」都很困難，因為找不到練習的方向，其中挫折往往讓人忍不住放棄。因此我們欲利用演算法輔以人工智慧來解決這個問題。藉由程式分析使用者寫出的字，找出使用者字跡的問題所在。讓使用者了解自己的字跡有哪些不足，並給予簡單且有效的練習方針，讓使用者能順利地寫出「工整」甚至「漂亮」的字體。

1.2 研究動機

隨著科技日新月異，學習的過程越加便利，需要寫字的機會越來越少。導致年輕人的字普遍不如老一輩，甚至可說是相去甚遠。而我們身在這樣便利的學習環境中，這個問題我們也有深刻的體悟。我們都曾多次嘗試將字跡變得工整，但每次的練習始終找不到方向，練習後的結果也差強人意。因此我們認為將字跡練好可以分為三個部份：1.找出字跡的問題。2.正確的練習方向。3.大量的練習。

出於上述的原因，我們決定要設計一個包含上述功能，能幫助人們將字體練好的工具。這個工具要能夠讀取使用者的字跡，並藉由演算法輔以人工智慧分析使用者的字體，找出字體的問題，並在手機上及時給予使用者回饋。藉由這樣的方式提供想要將字練好的人們一個有效率的練習方向。以期許便利的科技不再是字跡潦草的主因，而是協助字體工整的一樣工具!

1.3 文獻回顧

字跡美醜相關文獻：

鄭裕篤在 2009 發表的論文[1]中將寫者字跡與標楷字體進行比較分析，如用點、線段、區塊重心、覆蓋範圍比對等方式來辨識寫字的美醜。此方式證明了字跡的美醜與結構成正相關。根據 Simonnet, Anquetil, 和 Bouillon 在 2017 年對手寫法文字母的研究[2]中指出，字的形狀、筆畫順序、傾斜方向都是決定字跡品質的因數。在對字形的研究中使用的方式類似鄭裕篤使用的空間覆蓋法，由此可看出不論是中西文字都需要考慮字跡在空間中的分布情形。

而電腦字體設計師曾國榕(Go-Rong Tseng)先生在「中文字的重心」[3]一文中指出，字體協調的原理包括平衡、均間以及其比例；平衡可用書法中的九空格為例，可以檢視上下、左右的重心結構。而文中提到了**視覺重心**一詞，是一種字與字之間排序後的重心概念，同時也點明了字體排序的目標並非直覺的

上下對齊而是重心的對齊。

由此可知，重心可作為改善字跡的核心目標，不論是單一字元或是整行文字都是如此。本系統也運用此概念，藉由程式將寫字時不可見的重心位置標示出來，讓使用者能夠對字體的結構更有概念，同時藉由重心的偏移找出字跡的缺陷並得以改善。

字元辨識相關文獻：

字元辨識是影像處理[4]的一個分支，可以想像成對一個包含字跡的圖片進行運算，從而得到想要的資料和結果。大致可以分為以下步驟：1.輸入字跡圖片。2.由硬體進行字元取樣。3.前期預處理包括：去背、降躁去雜訊。4.最後將圖片進行辨識並且輸出結果，詳細步驟如下方圖 1。現在廣為使用的光學字元辨識（Optical Character Recognition，OCR）[5]，是藉由光學反射的原理，將文本上字元進行取樣，利用文本具有對齊性(字元長寬相等)的特性逐行掃描，將一個個字元存檔以利後續運算，這個過程稱之為轉檔。而後只需要與資料庫比對便可得出是何種字體並輸出對應的編碼。

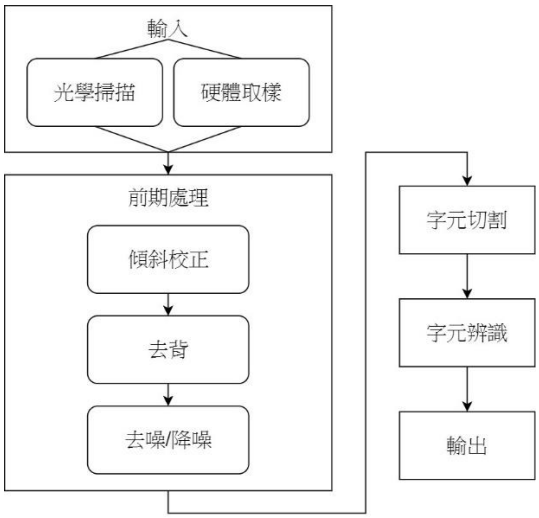

圖 1：字元辨識流程

目前字元的辨識採用的是機器學習的方式，在資料庫中輸入大量的資料建立類別，將取出的字元與資料庫進行比對找出相似度最高的結果並輸出。目前深度學習大致上可分成 CNN (捲積神經網路)[6]、RNN （循環神經網路）[7]、GAN（生成式對抗網路）[8]等類別。CNN 在處理空間上連續的資料較為突出；RNN 會處理有時間序列、語意結構的資料；GAN 則可讓電腦看到自己做過甚麼樣的事情，例如作畫時若不使用 GAN 可能導致電腦在畫人時出現兩顆頭，而使用 GAN 網路電腦在畫過一顆頭後會知道這件事已經做過，避免畫錯。

簡易的字元辨識模型多使用 CNN（捲積神經網路），是因為字元辨識需要對一資料矩陣進行運算，會需要使用二維的捲積（Spatial Convolution）。在訓

練模型時為了提升模型的精準度，可增加神經網路的隱藏層[9]使圖片的特徵能夠被細分；增加丟棄層（Dropout layer）[10]可以使輸出結果正規化，避免擬合過度（Overfitting）。

網路傳輸相關文獻：

HTTP 持久連結[11]（HTTP Persistent Connection，簡稱 HTTP Connection），是一種使用基於**傳輸控制協定**（Transmission Control Protocol，簡稱 TCP）[12]的網路連接協定。其特點是使用同一個 TCP 連接來傳送與接收多個 HTTP 請求，而不需為每一個請求打開新的連結。TCP 是一種可靠的資料傳輸連接，在傳遞封包時會給予每個封包序號，若發生序號遺失則重新發送該序號封包，以確保不發生封包遺失的情形。相較於 TCP 的高可靠度，能接受封包缺失的**用戶資料包協定**（User Datagram Protocol，簡稱 UDP）[13]也是廣為使用的一種通訊協定，常用於影像或音檔的資料傳輸，由於這類的檔案資料量龐大，缺失封包的數量較高，但對整體影像的影響極小，所以可以藉由忽略缺失的封包換取傳輸的時間和流暢性。字元辨識中若要使用 Wi-Fi 進行傳輸，其資料是不可缺失的。因為資料傳輸的過程中若遺失的是特徵位元（具有意義的位元），則會導致字元辨識的嚴重誤差。故採用基於 TCP 的 HTTP Connection 來建構傳輸路徑。

1.4 專題貢獻

現今提供協助人們練字的工具甚少，常見的練字工具也無法提高練習的效率。就算參考了他人的練字心得，也不見得適合使用者遇到的情形。本系統認為想要提高練字的效率，首先得讓使用者了解自己字的不足。了解自身問題後使用者便可以對症下藥，以最短的時間改善字跡的問題。這種切入角度大大的減少初學者的撞牆期，讓初學者不再因為找不到方向而放棄練習。基於此核心概念，本專題考量了現有的練字方式（ex.練字本、網路教學、練字 APP）各自的缺點，擷取了各別的優點並整合出全新的練字方式。

在硬體的部份，使用電阻式手寫板讓使用者透過實際的紙筆進行書寫，改善了光學取樣的不便性、失真問題，也避免了使用電容式感測的書寫環境不適的情況。

在軟體部份，本系統藉由使用者字跡的結構，提供了一個可靠且付予彈性的評分，再輔以圖形的回饋形式讓使用者能夠清楚地找到問題。使用者可根據自身書寫習慣進行調整，再藉由本系統的評分進行驗證。這種方式也改善了以臨摹來練字時只能練習一種字體的缺點，各種字跡在本系統上都具備可靠的評分比較，作為改善的方向。

深度學習模型的部分主要是要辨識出使用者所書寫的字為何字，因為系統必須依據使用者是寫什麼字給正確的回饋，所以模型的主要貢獻為提升系統回饋的可靠度以及正確辨識出使用者所書寫的文字。

第二章　理論背景

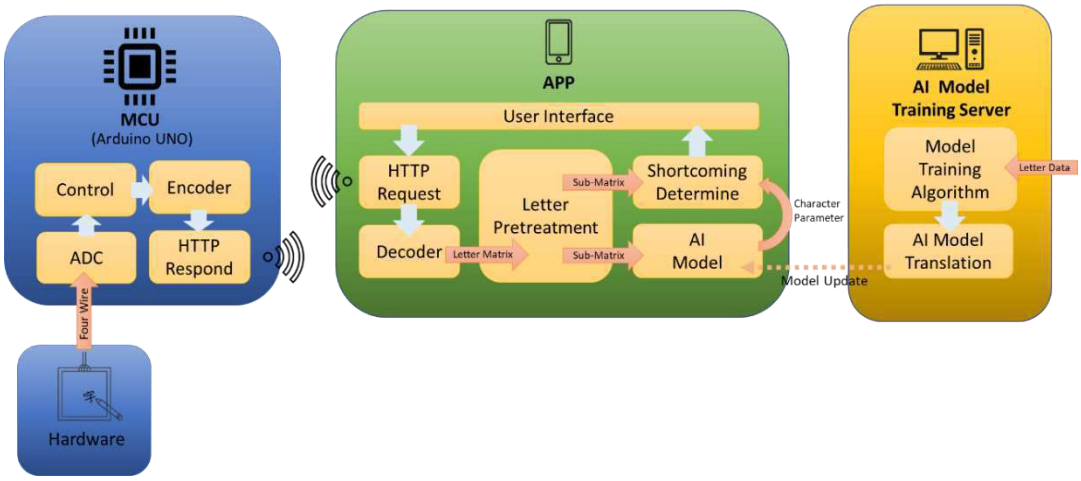

圖 2：系統架構圖

本系統架構如上圖 2 所示，由三大區塊所構成：硬體及微控器（藍色）、手機應用程式（綠色）、機械學習模型訓練（黃色）。雲端辨識模組與資料嵌入在應用程式內，硬體與 APP 則藉由 Wi-Fi 進行通訊。

本系統各部份的議題如下。

硬體的部份主要負責字跡的取樣以及資料的傳輸，分別遇到 **1.字跡取樣失真**、**2.傳輸時間過長**的問題。首先是取樣的失真，主要是因為精度使得取樣的結果無法正確的對應到 512*512 bitmap 上。再來則是在傳輸時間過長，因為資料量過大的造成的。由於資料維度會決定取樣的精準度，無法降維只能考量資料的壓縮，因此如何在硬體運算能力有限的情形進行最大的壓縮是硬體傳輸的重要議題。

應用程式為整體系統的整合端，應用程式最主要的技術議題分別是 **1.字跡圈選**和 **2.缺陷分析**。字跡圈選為本系統中最核心的一個步驟，如何快速且正確的將字跡圈選出來是演算法需要思考的，難度在於要從硬體取樣後具備雜訊和字跡失真的資料中將字跡取出並還原。而缺陷分析則是在使用者回饋前的重要步驟，同時也是量化字跡美醜的關鍵。這一步驟的議題在於如何在中文字中取出能決定好壞並能適用於所有中文字系統的特徵參數，並給予可靠的問題回饋。

深度學習模型主要的研究方向為 **1.提升模型辨識率**和 **2.改善 overfitting** 的問題。由於本系統是以繁體中文字作為主要訓練目標，而繁體中文圖片的複雜度相較於簡體文字以及英文高，在模型訓練時會隨著訓練次數的增加而更容易造成 overfitting 的問題，具體的表現為在訓練時的辨識率極高，但在實際測試另一批資料時，卻容易發生辨識錯誤的問題。而 overfitting 也可能是由 Activation Function 造成，因為資料的分佈通常都不是常態分佈，所以特徵數值都會落在飽和區域內導致越深層的網路對於特徵數值越不敏感 進而產生 overfitting 以及梯度消失等問題。

硬體設計

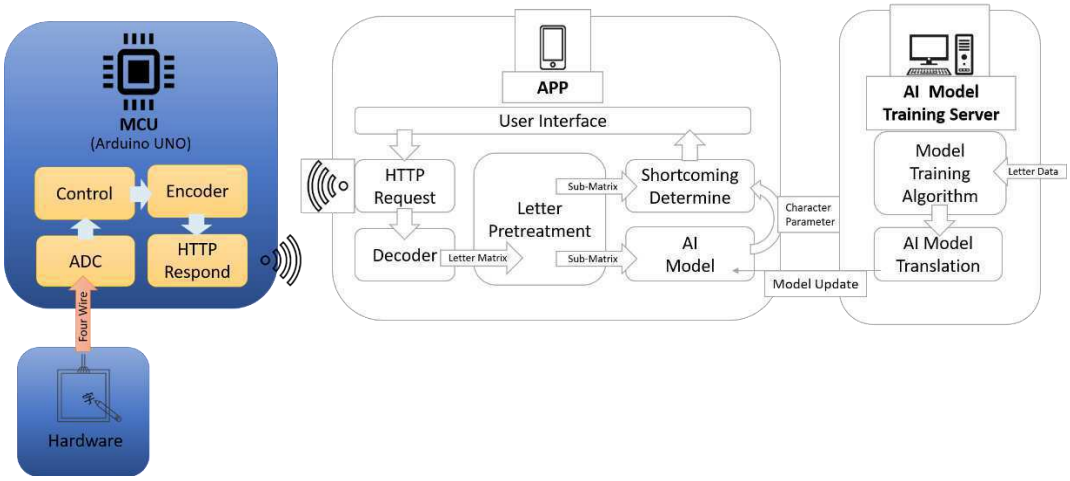

圖 3：硬體-系統架構圖

(一) 概述

圖 3 為硬體及微控制器，其主要負責：電阻屏接收字跡、接收應用程式之 HTTP Request、字跡資料壓縮、數據傳送。硬體區塊下方為電阻式手寫板，上方 MCU 區由微控制器(Arduino）與 Wi-Fi Shield 組成，類比數位轉換器(ADC)負責字跡取樣，控制器結合 Wi-Fi Shield 建構資料傳輸(HTTP Server)功能，Bitmap 壓縮部分(compression)目的是確保資料可在特定時間內傳完。

(二) 內容

a. 下方硬體區塊

此硬體區塊為電阻式手寫板，如圖 4、圖 5 的實體圖示，用來讀取使用者手寫字跡。當計算 Y 座標時，在 Y+電極施加驅動電壓，Y-接地，由 X+測量接觸點電壓，觸點電壓與驅動電壓比等於觸點 Y 座標與屏高度之比(1) (圖 6)。計算 X 座標原理同 Y 座標，觸點電壓與驅動電壓比等於觸點 X 座標與屏寬度之比(2)(圖 7)。

$$y_{ADC} = \frac{V_{x+}}{V_{drive}} \times height_{screen} \quad (1)$$

$$x_{ADC} = \frac{V_{y+}}{V_{drive}} \times width_{screen} \quad (2)$$

因 Arduino ADC 輸入為 0 至 1023 的數值，但本系統所紀錄為 512*512 的 bitmap，因此須將類比轉換器讀入的數值做轉換，同時需考慮電阻屏的長寬比例，才可使讀取的字跡等同寫入的字跡比例。定義 X 軸範圍 0~512，並運用比例關係$\frac{y}{512} = \frac{height_{screen}}{width_{screen}}$，找出 Y 軸範圍為 0~411。因此實際儲存座標需經過以下(3)(4)運算。

$$x_{real} = \frac{V_{y+}}{V_{drive}} \times 512 \quad (3)$$

$$y_{real} = \frac{V_{x+}}{V_{drive}} \times 412 \quad (4)$$

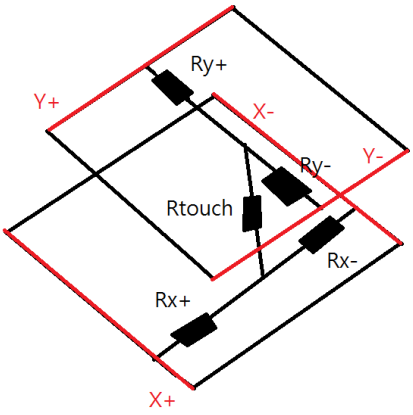

圖 4：電阻式手寫板

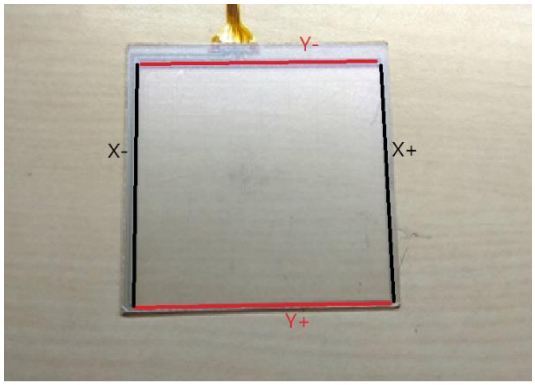

圖 5：電阻式手寫板

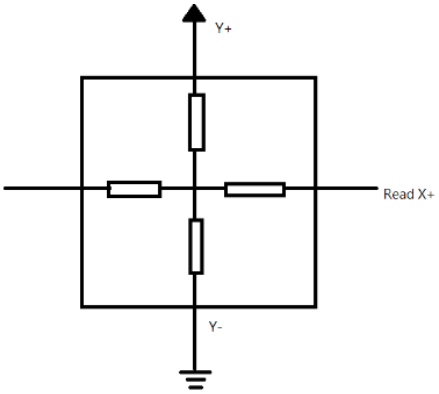

圖 6：Y 座標量測

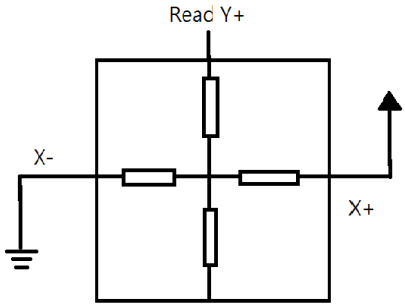

圖 7：X 座標量測

b. 上方 MCU 區塊

本微處理器(Microcontroler Unit)為主控版，負責整體硬體控制、數據收集、網絡通訊、為硬體系統主要中心。本區塊由 Arduino 及 Wi-Fi Shield 組成，其簡易的開發環境及經濟實惠的價格成為選用理由。主控版內部控制程式，根據設計流程控制周邊硬體設備、進行數據傳輸。Arduino 將工作分為三個任務：**手寫資料收集**、**網路通訊傳輸**、**Bitmap 壓縮**。

i. 手寫資料收集

手寫資料收集會持續性監測壓力是否大於閥值，當壓力值大於閥值時，表示使用者正在書寫，即將讀取到的位置點記錄到 512*512 bitmap 中。類比數位轉換器(Analog-to-Digital Converter，ADC)為 Arduino 主控板內建設備，其擁有 10 位元的解析度(0~1023)用於轉換電阻式手寫板的類比訊號至數位數據。

ii. 網路通訊傳輸

在與手機應用程式通訊方面，由 HTTP 作為通訊協定，當手機端透過 Wi-Fi 頻道傳送 HTTP Request 給 MCU，MCU 會進行 HTTP Respond，並將資料放入 HTTP Body 中。

iii. Bitmap 壓縮

Bitmap 壓縮是為了可以讓使用者減少傳輸所需的等待時間，因此將 Bitmap 進行壓縮，以減少傳輸資料，節省傳輸時間。而壓縮方法使用三種不同方法進行測試，分別為 Run-length encoding、Huffman coding、Rice coding。

(1) Run-length encoding[14]

由於本系統使用 HTTP 協定，需使用 ASCII code 進行傳輸。為避免使用到 ASCII code 之保留字元，故使用 6bit 進行編碼，第 6bit 為符號(1,0)，後 5bits 為此符號的連續數，前 2bits 固定為 0 可用於檢測壓縮是否錯誤。此壓縮對於連續資料有很大的壓縮效率。

(2) Huffman coding[15]

本系統是以 4bit 為基礎，設計出對應的 Huffman 編碼，在設計此編碼前，需先設計 Huffman Tree，由於使用的是固定的資料作編碼，而非每一資料製造另一種編碼，編出來結果在不同的資料下使用，大小雖然差不多，但卻不及 Run-length 好。

(3) Rice coding[16]

此編碼方式是由讀入數值的商和餘數所組成，使用固定的除數(M=16)，商為讀入數值(本系統一次讀入 2byte)除 M，商使用 unary 方式編碼(即先用商數目個 1 後接 0)，再接餘數(由於除數可為 2 的指數次，因此餘數即用此指數次的 bit 數表示)4bit，此編碼方式如遇到讀入為較大數值時，則會使編碼大小大於原資料大小。

壓縮方式	Run-length encoding	Huffman coding	Rice coding
壓縮執行時間	0.9s	0.9s	1.1s
傳輸所需時間	4.9s	7.1s	7.2s
壓縮比	3.85	2.463	0.746

表 1"施"字 Bitmap 資料的壓縮比較

壓縮方式	Run-length encoding	Huffman coding	Rice coding
壓縮執行時間	0.9s	0.9s	1.1s
傳輸所需時間	4.7s	7.9s	8.6s
壓縮比	3.795	2.447	0.7

表 2 "劉"字 Bitmap 資料的壓縮比較

壓縮方式	Run-length encoding	Huffman coding	Rice coding
壓縮執行時間	0.9s	0.9s	0.9s
傳輸所需時間	4.6s	7.6s	9.8s
壓縮比	3.85	2.46	0.925

表 3 "陳"字 Bitmap 資料的壓縮比較

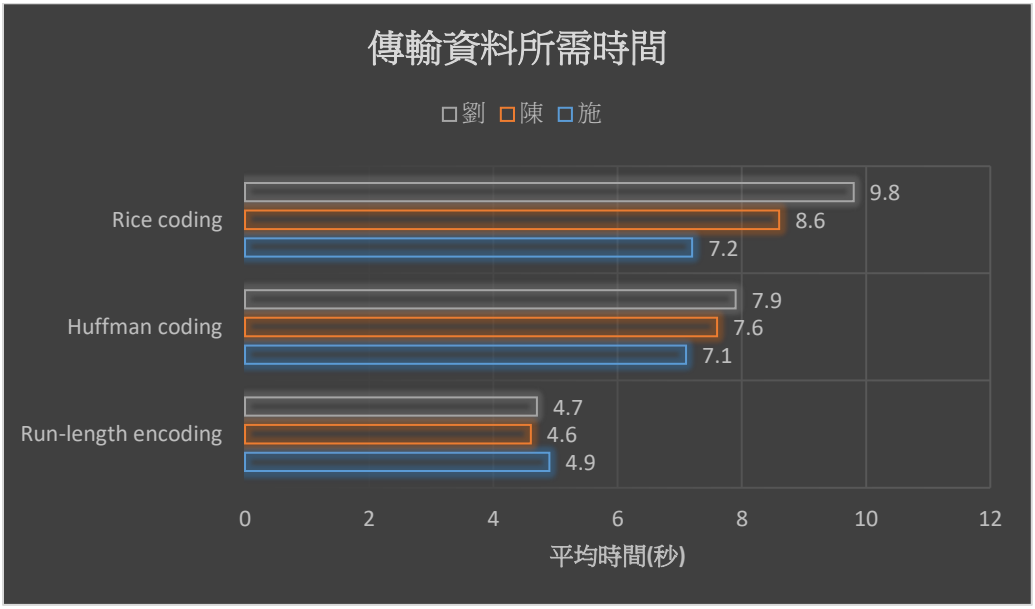

圖 8：傳輸時間比較圖

如上方所提到的使用三種不同壓縮方式進行測試，分別測試三種不同的資料(表 1 施、表 2 劉、表 3 陳)，紀錄個別資料的壓縮執行時間、傳輸所需時間、壓縮比等參數。

由圖 8 中可看出，遊程編碼壓縮後資料量最少，且壓縮所需時間也較少，當壓縮後的資料越少，則傳輸所需時間越少，因此可在 10 秒內傳完此資料。網路通訊傳輸也是持續監控是否接收到手機應用端的請求，若有則回覆給手機應用端並發送新的資料。

手機應用程式

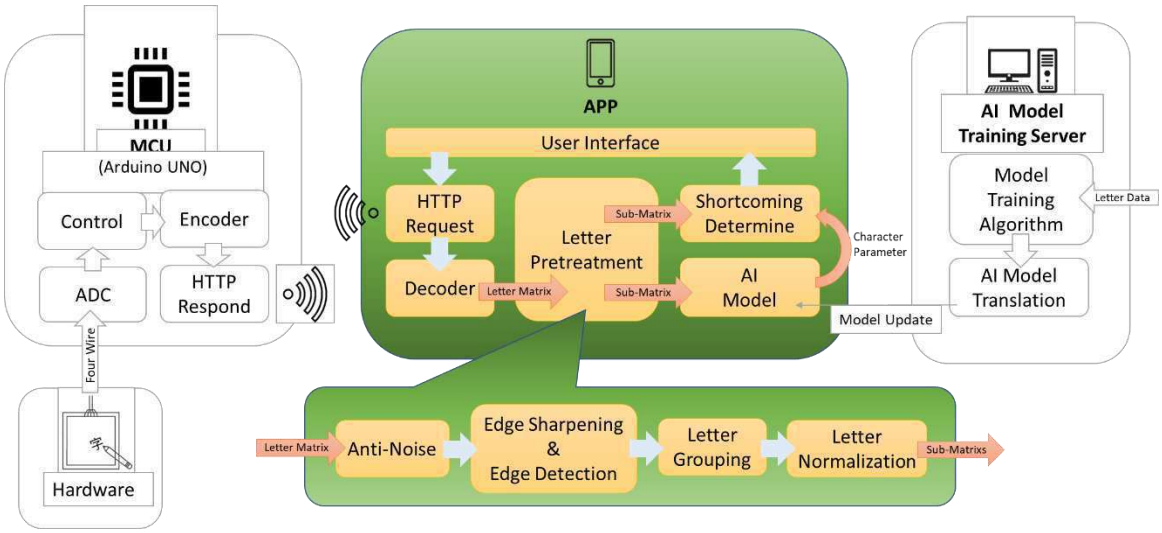

圖 9：APP-系統架構圖

(一) 概述

圖 9 為應用程式所實踐之演算法，主要負責串聯整體系統運作。主要負責接收硬體傳送之使用者字跡、從字跡陣列中各別取出所有使用者所寫字體、將字跡送入辨識模型內辨識、針對辨識結果取出對應特徵參數、搭配特徵參數計算使用者字跡缺陷、針對缺陷給予使用者回饋。

其中演算法有：UI 介面設計(User Interface)、HTTP 通訊協定架構(HTTP Connection)、字跡陣列解碼(Letter Decoder)、字跡預處理(Letter Pretreatment)、嵌入辨識模型(AI-Model Build-In)、字跡缺陷計算(Shortcoming Determine)、回饋使用者(Feed-Back)。

字跡預處理由：去背(Image Matting)[17]、抗噪(Anti-Noise)[18]、邊緣銳化與邊緣檢測(Edge Sharpening & Detection)[19]、字跡群組化(Letter Grouping)、字跡正規化(Letter Normalization)等演算法組成。

字跡缺陷計算由：十點重心計算(Center of Gravity)、數據正規化(Data Normalization)、線性回歸(Linear Regression)、問題分類(Problem Switch)等演算法組成。

(二) 流程

如圖 10 所示，使用者開啟後進入首頁後可選擇執行測試模式或示範模式進入不同的第二頁，差別在於資料的來源不同。測試模式的字跡是來自使用者在硬體書寫後，經由 Wi-Fi 傳輸並解碼後取得；示範模式則是藉由讀取事先建置在應用程式內的字跡圖檔來繼續後續運算。

取得字跡矩陣後會進入負責圈選字跡的第三頁，此頁面會先顯示未轉換前的字跡矩陣，並在使用者按下按鈕後將字跡圈選出來。

圈選完成後使用者將會進入第四頁，第四頁會將每個被圈選出的字跡獨立出來，並顯示字跡內的參數。每個字跡都有對應的按鈕可以進行字元辨識，辨識完成後會取得特徵參數並計算使用者字跡缺陷。而後進入第五頁，第五頁內會顯示該字元的缺陷和改善建議。

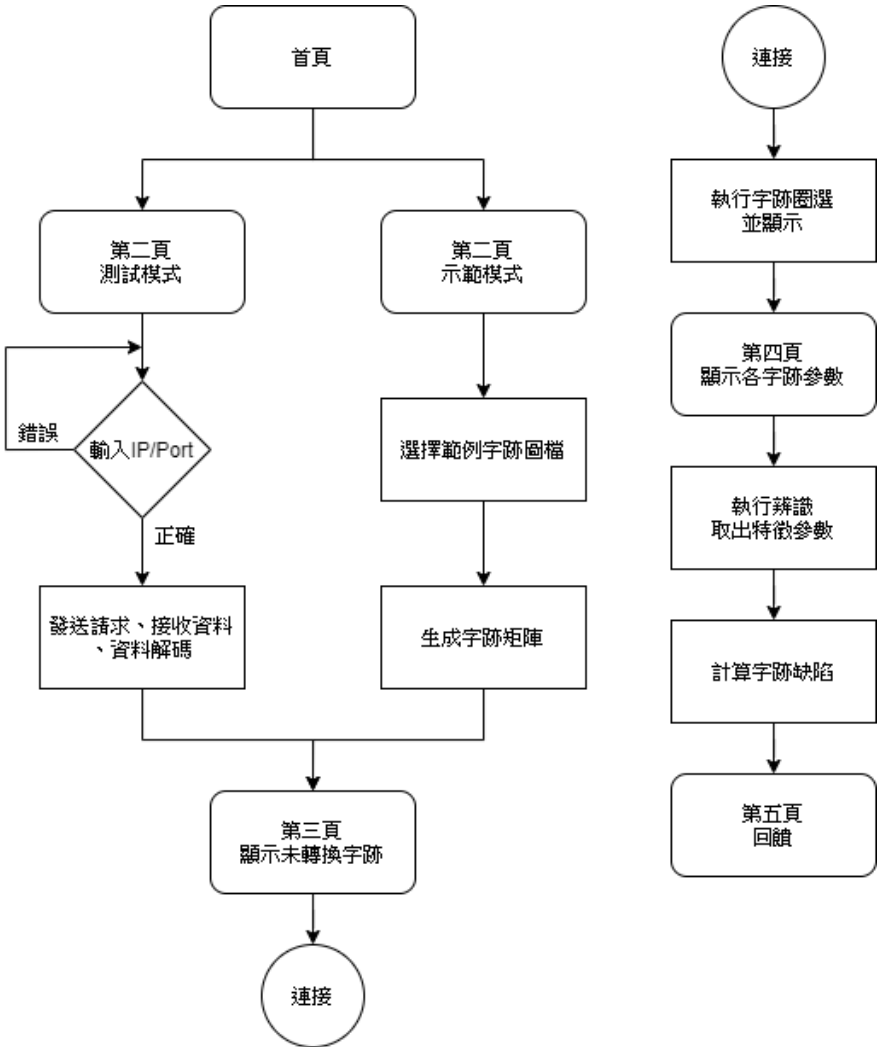

圖 10：APP 流程

(三) 演算法介紹

a. 資料傳輸方式

透過 Wi-Fi 使用 TCP 傳輸層的 HTTP 協定和硬體連線，基於 HTTP 協定的特性，客戶端(APP)不需與伺服端(MCU)事先建立傳輸通道，僅在需要資料時進入對應的 IP 位置即可。

b. 資料預處理

應用程式解碼完成後將取得使用者手寫的原始筆跡，但這筆資料尚需經過以下步驟方能將個別字元拆開。

i. 資料去背

硬體手寫板輸入的過程中，容易在手寫的過程中碰觸到外圍輪廓，使得字跡矩陣(Letter Matrix)的外圍輪廓出現特徵點。此現象會導致空間卷積(Spatial Convolution)出現溢位(Over Dimension)的問題。因此在運算前需將出現在外圍輪廓的特徵點消除(設置為零)。

ii. 雜訊點消除(抗噪)

在使用者書寫的過程中，常會因某些個人的習慣而留下非字元的字跡，這類不具意義的字跡稱為雜訊點(Noise)。若不處理這類的雜訊點，輕則增加演算法計算量，嚴重則可能導致 1.字跡圈選錯誤 2.字元辨識錯誤，使得系統可靠度降低。

我們可以將這類的雜訊點分為獨立和相依，若在後續演算法(斷點補償與字跡群組化)中該雜訊點會被字跡或其他雜訊點合併，稱為相依；反之則為獨立。而為了避免消除雜訊的過程中消除到使用者的字跡，演算法定義一消除門檻，在此步驟以消除獨立雜訊點為優先。由於消除門檻的搜尋半徑遠小於群組化的搜尋半徑，許多相依雜訊點在此會被視為獨立一併被消除。

iii. 斷點補償

由於硬體取樣頻率(Sampling Rate)的頻率不足，導致原本連續的筆劃會分成許多離散的斷點。這個問題有兩個解法，其一為更改硬體的儲存格，將資料依照先後順序儲存；其二便是在應用程式內藉由演算法進行斷點補償。由於前者會增加硬體負擔，故本系統採用斷點補償。本系統是利用邊緣銳化(Edge Sharpening)的會外擴的特性將離散點外擴，以此方法使資料達到片段連續來實踐補償。

邊緣銳化的方式有許多種，如 Roberts、Sobel、Laplacian、Marr-Hildreth、…等，本系統使用 Laplacian of Gaussian[20] (簡稱 LoG)，此類方法雖優於上述前三種方式，但仍不是最佳的邊緣測試法。不過其結果已經達到了本系統的要求，且系統所需為其外擴特性，因此不再加強邊緣銳化的精度。結果如下圖 11、圖 12。

LoG 其原理是對一範圍內的數值進行加權運算，藉此取得該範圍的梯度(Gradient)，若梯度超過標準梯度 σ 則將該點標示為 1。此類方法在訊號變換幅度不大(Smooth)時的效果會極差，但本次系統的資料為 1_bit 的 1/0 離散訊號，故此特性不影響邊緣銳化的結果。

圖 11：斷點補償前

圖 12：斷點補償後

iv. 資料輪廓化

此步驟其實與上一步驟是同時完成的，利用邊緣銳化形成的輪廓進行下一步驟的群組化。值得一提的是此外擴的大小取決於空間卷積的維度，若維度太小會導致斷點補償的效果不佳；太大則造成使用者字跡失真。資料連續性越高，後續群組化的計算量會越小，成平方關係。這也是為何前面要先消除雜訊點的原因，每多一個雜訊點都會使計算時間大大增加。

v. 字跡群組化

由於中文字不像英文或符號筆劃完全連續，中文字中筆劃有許多片段連續線段，且本系統支援在任意位置書寫任意數量的字元，字跡圈選的難度也大大上升。故本系統放棄一次到位的圈選方式，改分成以下三個步驟。

(1) 連續字跡群組化

下方圖 13 為群組化之流程圖，先對整個字跡矩陣做行列掃描，但凡遇到尚未群組化字跡訊號，則成立一群組紀錄其座標邊界並搜尋其範圍。若其範圍內具有其他字跡訊號表示連續，更新群組座標邊界並將位置轉移至該訊號，不斷重複直到回到原點。

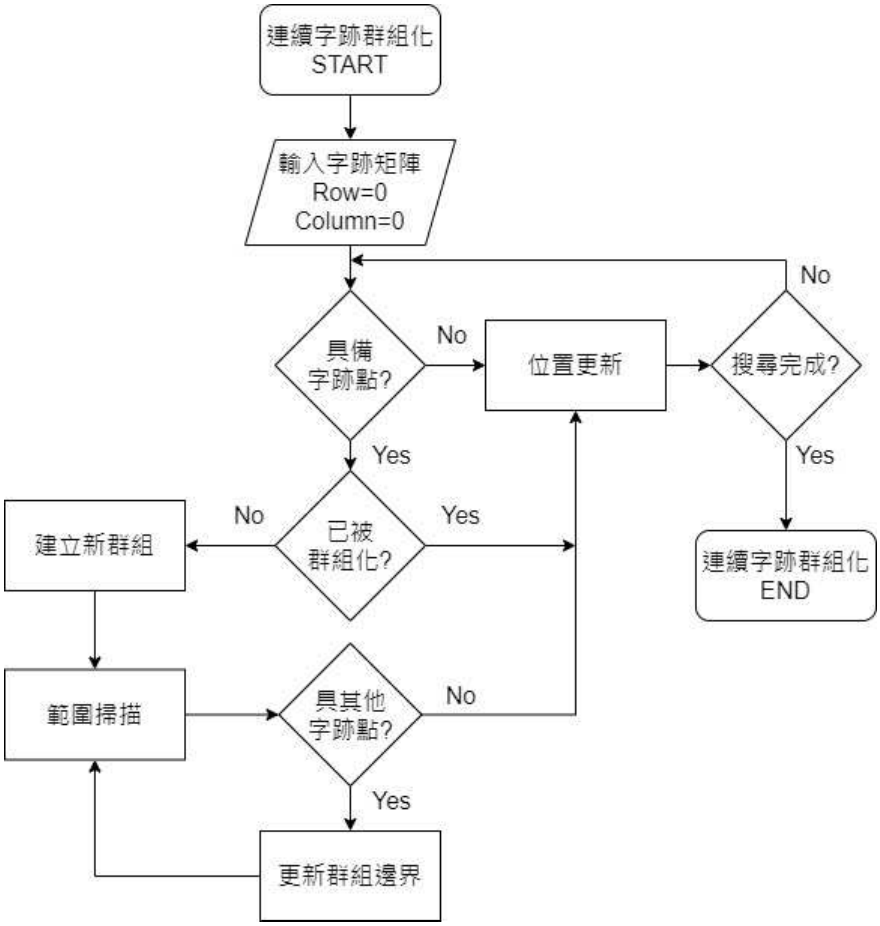

圖 13：連續字跡群組化流程圖

(2) 群組搜尋與合併

經過上一步驟所有字跡點與雜訊點都有了自己的群組。本系統利用中文字方正的特性設計搜尋半徑的算法，藉此賦予每個群組各自的搜尋半徑。若兩個群組的搜尋半徑重疊，代表這兩群組之間有相依性，則進行合併。合併後的新群組會獲得新的搜尋半徑，並繼續參與合併，直到所有群組互相獨立，表示圈選完成。

若群組數量為 n 個，此演算法的執行次數至少為$\frac{n*(n+1)}{2}$次。這也是為何我們會在此前先消除過一次雜訊點，因為每個雜訊點都會自成一個群組，大大增加運算時間。

(3) 雜訊群組移除

之前有提到過相依性高的雜訊點，在演算法中被視為字跡而無法消除。在此步驟我們可以藉由分析每個群組的平均長寬，找出這類雜訊點群組並將之移除。

vi. 字跡大小正規化

此步驟是用來將分割出的字跡轉換成字元辨識所需的格式，如維度、字跡置中等等。

c. 辨識模型嵌入

將 Tensorflow 訓練完的辨識模型(.tflite)嵌入應用程式內，使應用程式具備辨識字元的能力。

d. 計算字跡缺陷

此步驟又可細分為以下步驟

i. 十點重心計算

本系統對各種不同電腦字體進行重心分析，發現一個好看的字不僅是總重心置中，在多點重心也成立。中文字所說的上下切齊、左右對稱，也可在多點重心中看出端倪。

圖 14：標楷體十點重心範例

因此本系統定義了十個不同的重心，分別為：總重心、左上、中上、右上、左中、中心、右中、左下、中下、右下等十點。圖 14 為各字元十點重心的展示圖，可以看出其各自置中的特性。

ii. 資料正規化

本系統取用標楷體做為特徵參數來源，則後續判斷需符合其方正特性。然而使用者字跡長寬並不具此特性，故需將使用者字跡進行正規化，將其長寬值消去。同時為了更加有效率地引用特徵參數，在此將各重心點座標重新定義為從總重心座標出發的極座標向量表示法 $\{R\angle\theta\ ,0\leq R\leq 1\}$。

iii. 線性回歸

本系統藉由線性回歸計算字跡是否達到上下切齊、左右對稱。結果如圖 15。

令回歸線 $y = ax + b$ => $\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

$$A^TAX = A^TB$$

$$Where\ A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix}, x = \begin{bmatrix} a \\ b \end{bmatrix}, B = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

同乘 A^T 消除左零空間求解可得回歸線 a、b 參數。

圖 15：標楷體重心點回歸線

iv. 問題分類

上述步驟已將判斷所需參數處理完成，包含極座標重心點和線性回歸斜率。藉此兩參數可將字體結構問題量化，本系統取其中兩種回饋，分別是字跡傾斜問題和失真區塊，並根據誤差量給予字跡評分。字跡傾斜是藉由比較字跡斜率與標楷體斜率，若兩斜率差異過大則判定為傾斜；判斷失真是藉由比較字跡與標楷體的重心偏移，並將誤差最嚴重的部份標示出來。每一個重心點的偏移都會降低字跡的評分，若字跡超過 80 分則標示以達到工整的標準。

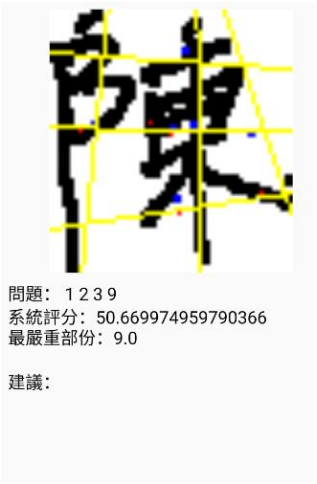

圖 16：回饋畫面

圖 17：偏移改善後

圖 16 是字跡回饋的頁面，此字跡略有左斜的問題，但最主要是有許多結構的失真(重心偏移)，分別是重心點 1(左上)、2(中上)、3(右上)以及 9(右下)。系統指出最嚴重的失真，意指若改善此區塊可最大幅度的改善字(如圖 17)。其餘誤差量較低的點與我們直觀判斷的結果略有出入，這也說明了為何練字會出現找不到方向的情形，直觀判斷的結果不見得是影響重心最主要的原因。

深度學習模型

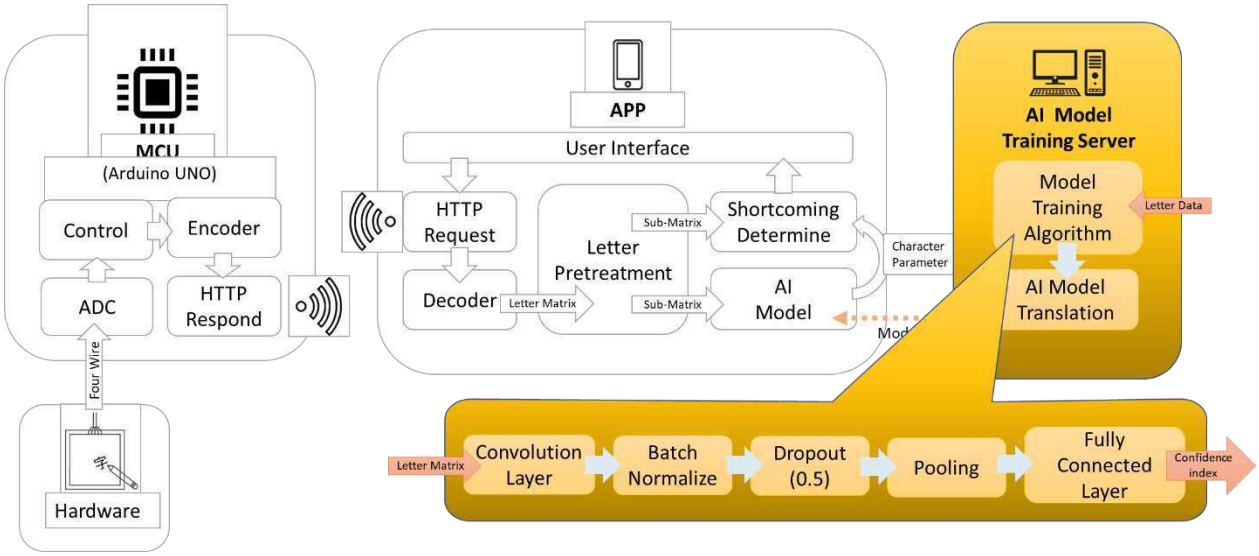

圖 18：雲端-系統架構圖

(一) 概述

如圖 18，深度學習的部分主要是透過捲積神經網路判斷使用者寫入的文字。字跡會因每個人寫字習慣的不同而有所不同，但是一個文字是否「工整」取決於字的筆順、空間對稱等等。本系統會將使用者書寫的文字與標準字（標楷體）做比較並且給予使用者改善其字跡的建議，所以判斷輸入文字是哪個字對整個系統來說是一個非常重要的依據，因為這個關係到使用者的改善回饋。為了要針對不同的使用者書寫的筆跡，本系統在數據蒐集時有特別加入文字部件間隔的較開、較緊湊、左傾、右傾、較大或者較小等等的文字使模型學習時可以因應各種情形作出準確的判斷。

(二) 流程

使用者書寫完後會生成一個 64x64 的文字矩陣(圖 19)，輸入深度學習的網路。

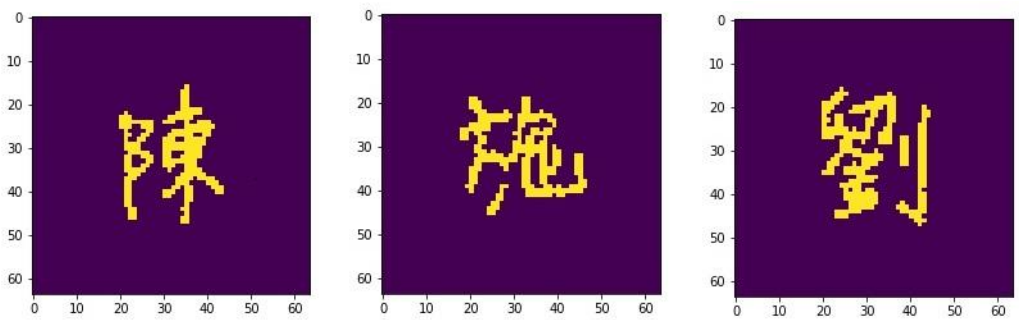

圖 19：字跡資料

圖 20：字跡辨識信心指數

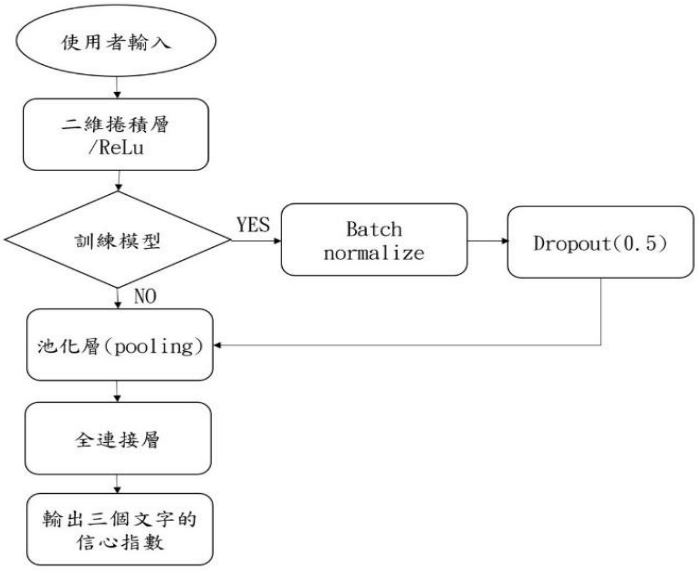

圖 21：模型訓練流程

神經網路中會有四層的捲積層（Convolution layer）後面再接上三層的全連接層（Full Connection layer），每一個捲積層在訓練時會先做二維捲積再放入激勵函數（Activation Function）中，訓練時通常只會加上一層丟棄層（Dropout）使一些神經節點被抽離，防止過擬合（overfitting）的情況發生，但本系統在 Dropout 之前再多加上一層批量規一化層（Batch Normalization）[21]，用以平均化 Activation Function 所生成的較極端的數據分佈，再送入池化層（Pooling layer）降低數據的維度。

最後將二維數據轉換成一維進入三層全連接層後輸出各個文字的信心指數。圖 20 是在 APP 上實際上寫三個文字經由深度學習模型所判斷出來的三個信心指數，若判斷該字不是此通道其信心指數就會極低。

整個流程如圖 21 所示，若在訓練階段才會進入 Batch Normalization 以及 Dropout 的部分；若是在測試階段為了保持資料完整性就需要保留所有的神經節點以得到真正的輸出結果。

(三) 模型詳細內容

a. overfitting 改善

```
start to train
0.1
0.1
0.1
0.1
0.1
0.1
Done
```

```
start to train
0.6759259
0.9907407
0.9907407
0.9907407
0.9907407
0.9814815
Done
```

```
start to train
0.30555555
0.8240741
0.9444444
0.962963
0.962963
0.9722222
Done
```

圖 22：各版本辨識率（左：第一版，中：第二版，右：第三版）

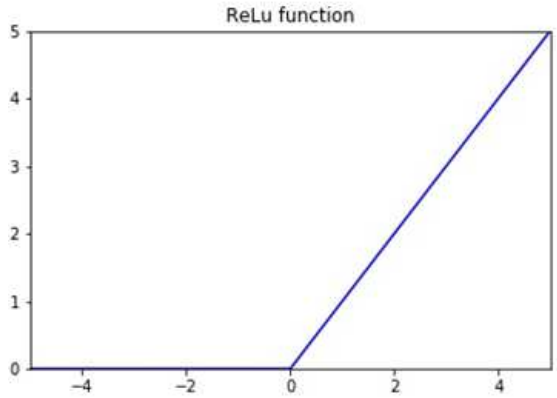

圖 23：Activation Function Curve

圖 22 左為本系統第一版測試的模型，只有三層捲積層（Convolutional layer）以及一層全連接層（Full Connection layer），每層都加上一個 Dropout(0.5)。其辨識率都只有 0.1 是因為測試資料維度過大，高達 512x512 且資料量只有 100 筆。

圖 22 中是本系統降低資料維度至 64x64 同時增加每種文字到 180 筆，共有 540 筆資料作為訓練資料。有 Dropout 卻出現 overfitting 的問題，發現是 Activation Function 出了問題，因為本系統此前使用的 Activation Function 是 ReLU（Rectified Linear Unit）函數[22]，其圖形如圖 23，如圖所示只要小於零就會歸零，造成經過此 Activation Function 之數據會非常極端，並非常態分布。

為了解決第二版由 Activation Function 所產生的 overfitting 問題，就是要加上前面所提到的 Batch Normalization，其原理是求出 Activation Function 所生成的數據集合 X 的變異數（Batch Variance）σ以及平均數（Batch Mean）μ其公式為：

$$X = \{x_1, x_2, \dots, x_m\} \tag{5}$$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{6}$$

$$\sigma = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \quad (7)$$

$$\hat{x}_i = \frac{x_i - \lambda}{\sqrt{\sigma^2 + \varepsilon}} \quad (8)$$

X 共有 m 個元素(5)，先求出平均數(6)與變異數(7)，再將這兩個數據帶入平均公式為(8)使原數據 X 的每個元數都可以正規化（Nomalization）。

在經由 scale and shift 的參數γ, β 使得所有元素都可以有常態分布而不會出現太過極端的數值，而γ, β也是要由神經網路訓練所得到的兩個參數，最終輸出的集合稱為 Y 為(9)而其元素的求法為(10)。

$$Y = \{y_1, y_2, \ldots, y_m\} \quad (9)$$

$$y_i = r\hat{x}_i + \beta \quad (10)$$

圖 22 右側透過加上 Batch Normalization 改善，辨識率與前兩版比較可以看出來 overfitting 的情況得以改善。

b. 深度學習模型比較

接下來為深度學習模型比較。當圖片維度不高時可以使用普通全連接層網路（FCN）來辨識即可，如果圖片過於複雜就可能會需要過捲積來精確的提取圖片特徵，以下將會以捲積神經網路（CNN）以及全連接層網路來調整參數比較其優劣，並且擇優使用。

圖 24 主要以調整 Dropout 的參數以及比較捲積神經網路再加上 Batch Normalization 之後的辨識率變化。可以看到圖 24 的紅線是 Dropout(0.7)的模型，其辨識率不超過 0.4 是因為模型捨棄過多的神經節點，預測都無法穩定，出現時高時低的情況且無法提高辨識率。綠線是只拿掉一半的神經節點的捲積神經網路，辨識率雖然有上升，但是在四十步與六十步之間辨識率由 0.604 提升到 0.955 之後就一直保持在 0.94 以上，這種情形就是前面提到的由 Activation Function 所產生的 overfitting。藍線是在 Activation Function 後在 Dropout 前加上 Batch Normalization，可以看的出來辨識率的曲線比只加 Dropout 的曲線還要緩和，在第四百步的時候辨識率達到 0.943，並且有在繼續上升的趨勢。

圖 25 為比較捲積神經網路和普通全連階層的優劣，虛線為未加上 Batch Normalization 的模型辨識率圖，實線為以加上 Batch Normalization 的模型辨識率圖，再藉由折線圖選擇較優秀的模型使用。

圖 25 的綠色以及綠色虛線是只有加上 Dropout(0.5)的模型，兩個模型的辨識率都在六十步之後達到 0.9 的辨識率，很明顯都已經 overfitting 了。藍色以及藍色虛線除了有 Dropout(0.5)之外，捲積神經網路又加上二維的 Batch Normalization，而普通全連接層也有加上一維的 Batch Normalization，可以看出

捲積神經網路的辨識率是高於普通全連接層，原因在於捲積層是用三維的核（Kernal）去提取矩陣的特徵數值，其中包括長、寬還有 RGB 的色彩通道；反觀普通全連接層在一開始就要處理 1x4096 的資料，且只依靠 Activation Function 並無法有效得到此矩陣的特徵數值，導致普通全連接層的辨識率上升地很緩慢。

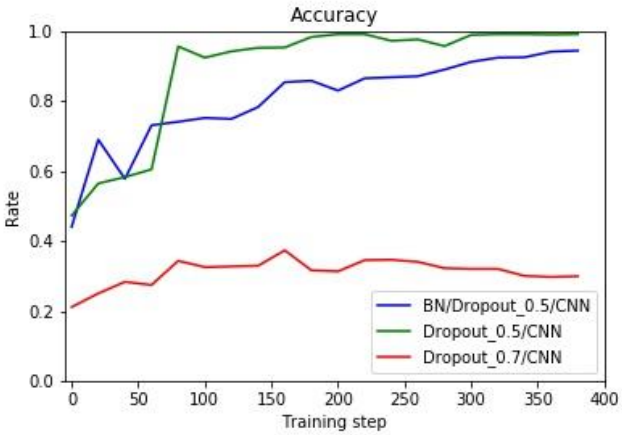

圖 24：CNN 辨識率折線圖

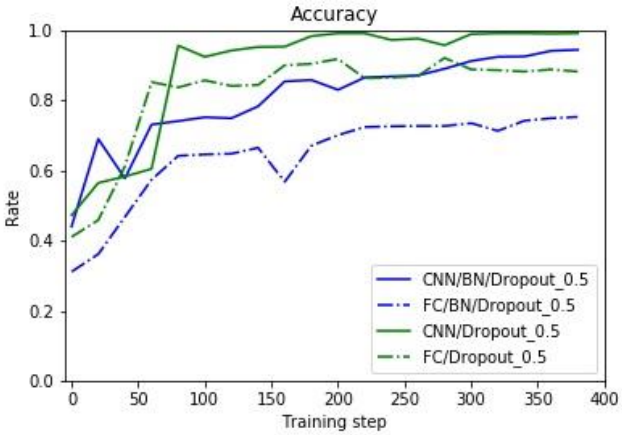

圖 25：捲積神經網路以及普通全連接層網路比較圖

第三章　實驗數據

功能展示：

下方四張圖片為本系統主要功能的展示，圖 26 為應用程式與硬體對傳的頁面，藉由輸入網址可以快速地取得使用者的字跡資料，應用程式會記憶前一次輸入的網址以便使用；圖 27 為字跡圈選畫面，圈選時間約在 1~2 秒左右視使用者書寫的字數決定；圖 28 會對字跡進行辨識，取出辨識後相對應的特徵參數並在此頁面同時計算重心(如圖 29)、斜率等參數以利後續回饋使用；圖 30 為系統對字跡的分析與評分，可藉由評分搭配圖形上的重心分佈與回歸線進行修正。

回饋展示：

圖 31、圖 32 是回饋頁面中的特殊狀況，圖 31 是當使用者字跡或間距過小造成字跡模糊的狀況；圖 32 是當字跡嚴重失真時的回饋，畫面中可以看到當結構為非方正的字體也會給予此回饋，原因是辨識模型尚未建入此類字體的特徵參數，當辨識模型擴充後即可修正。

圖 26：HTTP 連接

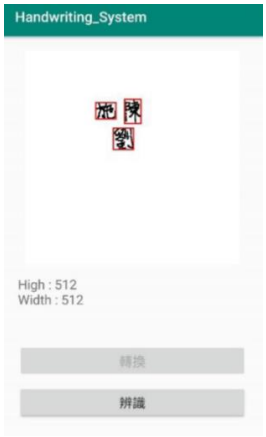

圖 27：字跡圈選

圖 28：字跡辨識

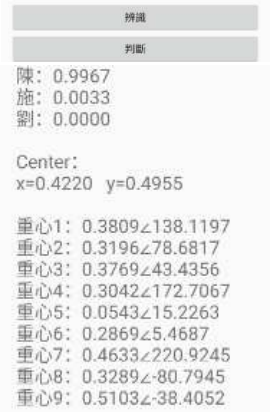

圖 29：十點重心

圖 30：字跡回饋

圖 31：模糊回饋

圖 32：結構失真回饋

軟硬體字跡對照：

下圖 33 為硬體與軟體之間的對照，可以看出畫面(a)是手寫板上的字跡，左側字跡對應到畫面(b)；右側字跡對應到畫面(c)。可以看出右側字跡較左側字跡來的工整，對應到應用程式的評分亦然。

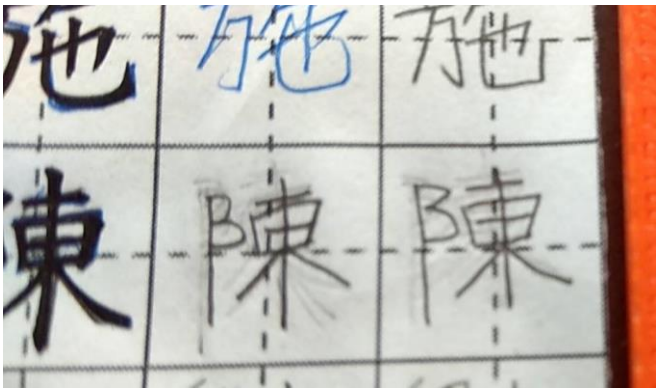

圖 33：(a)手寫板上字跡

(b)左側系統分析結果

(c)右側系統分析結果

未收錄字跡展示：

下方圖 34 為未收錄字跡的評分展示，可以看到左側字跡的評分是低於右側字跡的，這也是本系統藉由結構失真評分的優勢，即使字跡未收錄評分仍具有高低之間的落差。

圖 34：未收錄字跡:"李"字

第四章　建議與結論

影響字跡好壞的因素很多，人們熟知的姿勢、力道、筆順都是可能影響的變因。起初本系統以此方面著手，但隨著研究發現這些因素並非絕對，仍有人以不端正的姿勢、不一樣的筆順寫出「工整」甚至「漂亮」的筆跡。這導致本系統的開發一度失去方向。這項發現也代表著他人的練字心得不見得適用於各種情形，因為每個人書寫的習慣都是無法複製的。本系統認為這也是為何許多人不斷的臨摹和參考他人心得卻遲遲無法進步的主因。

因此本系統依據前文敘述之方式提供使用者一個觀察自己字跡結構的練字方式，藉此方式雖然不能真正給予使用者具體的改善建議，但也避免了讓使用者以不適合自身的建議進行練習，讓使用者可以自己調整自己的姿勢、力道、筆順，找出最適合自己的姿勢、力道、筆順等等，使系統更具彈性和貼合使用者。

在硬體的部份本系統選用了電阻屏進行取樣，優點是可以模擬出使用者熟悉的書寫環境，同時也不會發生光學掃描的角度、反光等失真。僅需墊在使用者的紙下便可以使用，與墊板的概念相符。硬體在傳輸的部份使用了 HTTP 的通訊協定，選用的理由是因為 HTTP 的連接方式簡單且為單向式，對使用者(應用程式)來說很容易操作，應用程式也無須額外負擔維持傳輸通道的運算成本。資料壓縮選用了 Run-length Encoding 的格式進行壓縮，其對於連續資料的高壓縮能力成為其選用原因。

在應用程式在資料整合的過程遇到了硬體取樣頻率不足的問題，由於硬體運算量已經飽和無法提升，因此在應用程式內藉由 LoG 邊緣銳化完成斷點補償。且本系統考量了使用者書寫的自由度問題，為使使用者能在電阻屏上任意位置寫上任意個字元，應用程式在字跡圈選的過程中捨棄了行列切割的特性改以鄰近搜尋的方式進行圈選。而在資料的回饋本系統使用了結構性的評分方式，輔以圖形化的字跡問題回饋，讓使用者能清楚地找到問題，並找到適合自身的改善方法。

辨識模型比較了 CNN 與 FCN 的辨識率，選擇 CNN 使用三維 Kernel 快速掃描特徵點，且辨識效率大於 FCN 的特性，用以提高訓練的效率與辨識的可靠度。在訓練過程中遇到了 Activation Function 飽和導致產生 overfitting 的問題，在加上 Batch Normalization 後校正資料分布，順利改善 overfitting 的問題，得以順利辨識出正確的結果。

第五章　未來發展方向

未來本系統可藉由增加辨識模組的頻寬提升可練習的種類，包含各式中文字、英文、符號等等，針對硬體元件可持續加強以下幾點：**(一) 改良手寫板書寫情況**：由於電阻式手寫板無法進行多點觸控。所以未來可用其他可多點觸控的手寫板進行優化。**(二) 硬體端及時回饋**：本系統回饋是於手機端進行顯示，使用者須從手機上觀察結果。若手寫板結合顯示板，可使系統各加便利，更可讓使用者在上次的字跡上進行練習。

針對軟體設計，可持續改善以下幾點：**(一) 段落字跡圈選：**目前字跡圈選尚未提供連字的切割，日後可根據平均面積方式進行延伸。**(二) 整行文字改善建議：**前文中曾提及影響字跡美觀度的要素其實是整體中心是否對齊，本系統可往此方面延伸以提供段落文字美觀度改善的建議。

第六章　時間進度表(甘特圖 Gantt Chart)

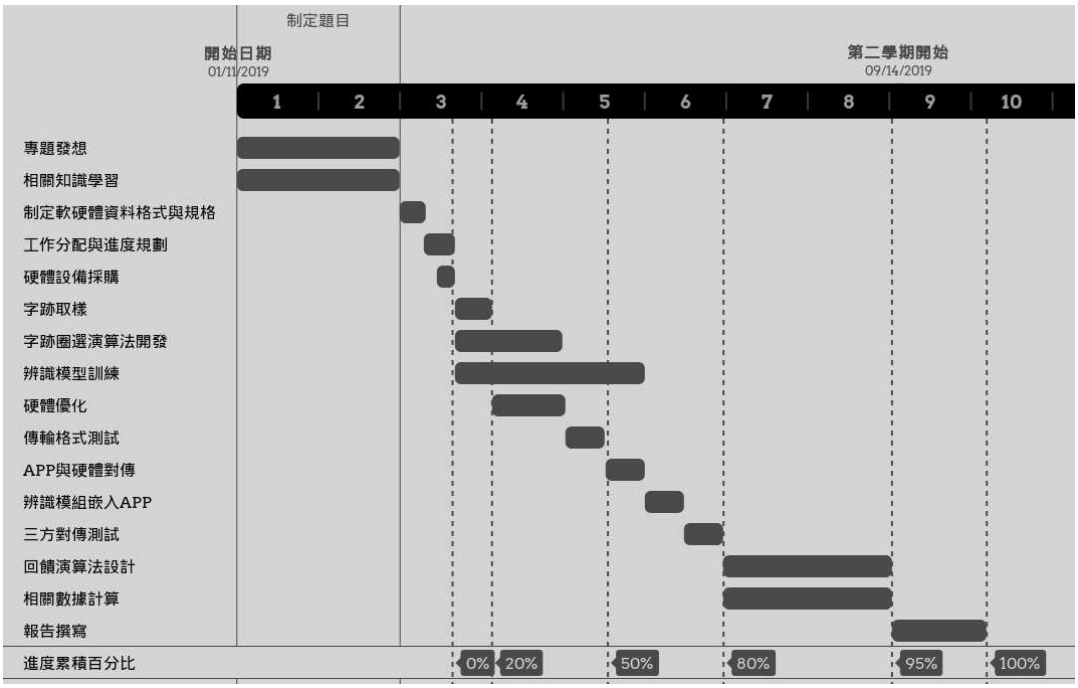

圖 35：系統甘特圖

第七章　參考資料

[1] 鄭裕篤（2009）。基於模糊測度之 Choquet 積分模式的網路書寫評量系統。國立臺中教育大學教育測驗統計研究所博士論文，台中市。

[2] Damien Simonnet, Eric Anquetil, 和 Manuel Bouillon (Sep 2017) "Multi-criteria handwriting quality analysis with online fuzzy models". Pattern Recognition 310-324

[3] 【字體心得】中文字的重心 2018 年 4 月 13 日。取自 https://medium.com/@tsenggorong/%E5%AD%97%E9%AB%94%E5%BF%83%E5%BE%97-%E4%B8%AD%E6%96%87%E5%AD%97%E7%9A%84%E9%87%8D%E5%BF%83-22c62091f371

[4] Digital image processing https://en.wikipedia.org/wiki/Digital_image_processing

[5] Optical Character Recognition https://en.wikipedia.org/wiki/Digital_image_processing

[6] CNN https://www.tensorflow.org/tutorials/images/cnn?fbclid=IwAR00HAYmLrqsqdSRerX7UxjMqkMt6WBhAl22marhw585jgx88UTb0ANEnh4

[7] RNN https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912

[8] GAN https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

[9] Steven Shen(2018)。改善 CNN 辨識率。https://medium.com/@syshen/%E6%94%B9%E5%96%84-cnn-%E8%BE%A8%E8%AD%98%E7%8E%87-dac9fce59b63

[10] Ray Lin(2018)。正則化神經網路 (2): Dropout&Others。https://medium.com/%E5%AD%B8%E4%BB%A5%E5%BB%A3%E6%89%8D/%E6%AD%A3%E5%89%87%E5%8C%96%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1-2-dropout-others-c35bb6f68dd7

[11] Margaret Rouse(2006).persistent connection (HTTP persistent connection). https://whatis.techtarget.com/definition/persistent-connection-HTTP-persistent-connection(December 23)

[12] TCP
Vinton G. Cerf; Robert E. Kahn (May 1974)."A Protocol for Packet Network Intercommunication". IEEE Transactions on Communications.(5), 637–648.

[13] UDP
Clark, M.P. (2003). Data Networks IP and the Internet, 1st ed. West Sussex, England: John Wiley & Sons Ltd.

[14] Run-length encoding
Mohammad Arif, R. S. Anand (2013) Run length encoding for speech data compression. 2012 IEEE International Conference on Computational Intelligence and Computing Research.

[15] Huffman Coding
Rabia Arshad, Adeel Saleem, Danista Khan(2016,Aug).Performance comparison of

Huffman Coding and Double Huffman Coding.Paper presented at 2016 Sixth International Conference on Innovative Computing Technology (INTECH),Dublin, Ireland
[16] Rice(Golomb) coding
Walter Daniel Leon-Salas(2015 July).Encoding compressive sensing measurements with Golomb-Rice codes.Paper presented at 2015 IEEE International Symposium on Circuits and Systems (ISCAS),Lisbon, Portugal
[17] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. (2001) A Bayesian Approach to Digital Matting. In Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR 2001), Vol. II, 264-271
[18] Hui KongA, Hatice Cinar Akakin and Sanjay E. Sarma (2013) Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications. IEEE Transactions on Cybernetics 43(6):1719-1733
[19] Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), 679-698.
[20] Optimization of Laplace of Gaussian (LoG) filter for enhanced edge detection: A new approach.
https://www.researchgate.net/publication/301402240_Optimization_of_Laplace_of_Gaussian_LoG_filter_for_enhanced_edge_detection_A_new_approach
[21] Batch normalization 原理與實踐
https://zhuanlan.zhihu.com/p/34879333
[22] 整流線性單位函數（ReLU）
https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

第八章　工作分配

姓名	工作項目
劉錕笙	應用程式設計 軟硬體通訊 字跡圈選 字體缺陷演算法設計與回饋
施丞祐	建立 HTTP Server、對傳資料壓縮 手寫類比訊號處理 硬體外部電路及包裝設計
陳儀銘	數據分析 神經網路訓練與嵌入

第九章　經費表

本系統使用材料包括電子零組件、電阻式手寫板、外部機殼等，總計約兩千元。