國立台北大學資訊工程學系專題報告

交通標誌即時翻譯系統

(Real-time Traffic Signs Translation System)

專題編號 : NTPUCSIE-108-005
專題組員: 林冠吾 410572011 李奕昕 410685006
羅紹豪 410685034 李憲騏 410685035
執行期間：2019 年 9 月 至 2020 年 6 月

1. 動機

台灣的土地面積狹小且道路繁雜眾多，伴隨的是路旁密密麻麻的交通號誌。在合法上路之前，要先經過駕照考試，其中就需要具備認得各種交通號誌的基礎能力。但是在實際上路後，看到號誌時不會像考試一樣有太多思考時間，而是需要在第一時間作出辨識，一次往往不只單一號誌，而是大量的號誌隨之來。這讓我們不禁思考，雖然我們身為台灣人長身處在這片土地，相對於常用的交通號誌能夠輕鬆判別，鮮少見到的號誌數量竟也不在話下，甚至還有些許號誌是我們時常容易搞混的；假如我們今天是位來自國外對當地不熟悉的自駕駕駛呢?先不論以上身為台灣人的我們就有的困擾，有些號誌上頭標示的不是圖案而是中文字，大多數外國人無法理解中文，便會造成無法正確判讀交通標誌，冥冥之中更增加行車的風險。

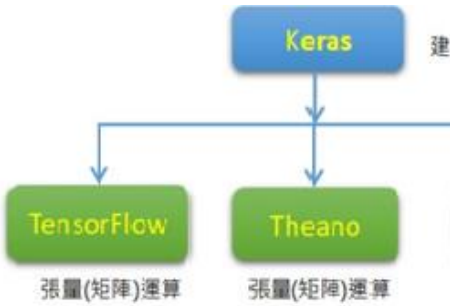
我們希望這個程式不僅能**正確判斷路邊的交通號誌**，更希望能在有充足的反應時間內，用**客製化的語音系統提醒駕駛者眼前出現的路標**。對於需要全神貫注於路況的駕駛來說，語音絕對是追求安全的第一選擇，能夠聽到自己熟悉的語言更是對駕駛的友善設計，聽到馬上能夠做出判斷，增進交通安全的福祉。

2. 環境以及相關技術

在專題的開發上我們使用 python 環境，進行第一版電腦版的實現，而後我們將電腦端的程式，移動到手機端上執行，以達到輕量性、便利性的使用，以下將先介紹我們使用到的環境:

A. Keras

Keras 是一個開放的原始碼，高階深度學習程式庫，使用 python 編寫，能夠運行在 tensorflow 或 Theano 之上，作者為 Google 的工程師，Keras 可以使用最少的程式碼，花費最少的時間，就可以建立深度學習模型，並進行訓練，評估準確率，此外 Keras 必須配合使用後端引擎(backend engine)進行運算，因此撰寫 Keras 只需專注於建立模型，底部的操作細節，如下圖所示



由此看出 keras 會將運算轉為相對指令，相當友善的程式庫。

B. Android studio

Android Studio 是 Google 於 2013 I/O 大會針對 Android 開發推出的新的開發工具，目前很多開源項目都已經在採用，Google 的更新速度也很快，另外 Android Studio 整合了 Gradle 構建工具，Gradle 是一個新的構建工具，Studio 亮相之處就支持 Gradle，可以說 Gradle 集合了 Ant 和 Maven 的優點，不管是配置、編譯都快速。

C. Opencv

OpenCV（Open Source Computer Vision）是一個包含許多電腦視覺相關演算處理的開放原始碼 Library，並且 OpenCV 可以在 Android、ios 上開發，支援的程式語言也有 C/C++，Java，Python，簡言之，opencv 是依用來處理圖像的工具，他讓電腦具備「看」的世界，而本專題使用於手機鏡頭的抓取與處理，並將相機的 frame 顯示在螢幕上。

而我們使用現有技術如下

A. 物體偵測(Object detection)

使用深度學習(deep learning)中的物體偵測技術，將圖片中有訓練的交通標誌判斷種類並框出物體位置，而我們使用的是 YOLO 深度學習網路，YOLO 網路是基於卷積類神經網路(CNN)的神經網路，是目前深度神經網路領域的發展主力，在圖片辨別上可以做到比人類還精準的程度，CNN 其運作方式如下:

(1) 卷積部份(Convolution)：卷積運算的 mask 又稱為 kernel map，一般的圖像會再加上激活函數(activation function)進行非線性轉換，得到特徵圖(feature map)，也就是說，它是通過在輸入圖像上滑動不同的卷積核並執行一定的運算而組成。

(2) 池化部份(Pooling)：池化層會對特徵圖進行非線性的降採樣(unsimpling)，其功能為不斷地減小資料的空間大小，因此參數的數量和計算量也會下降，但卻能保有特徵圖的特徵不便，一般常用「最大池化(max pooling)」其為將圖像切割成若干個矩形空間，並輸出每個子矩形空間的最大值。

介紹完 CNN 的運算後，接著來介紹基於 CNN 的類神經物體偵測方法，YOLO 深度學習網路，YOLO，對影像中的每一個位置，都只評估一次，所以又被稱 single-shot 方法，其判斷流程如下：

(1) 將影像切成 SXS 大小的格子(格子中若有物體則該格子會負責去偵測該物件)

(2) 每個格子會預測 B 個 bounding boxes 與 confidence scores 並同時預測 class(*Confidence scores 代表對應 bounding box 含有物件的信

心程度)，以及該 bounding box 中物體的精準度)

B. Socket

Socket 是網路上的通訊端點，使用者或是應用程式只要連結到 socket 便可以和網路上任何一個通訊端點進行連線，socket 之間的通訊就跟 process 間的通訊一樣。Socket 是一種識別碼，應用程式可用此唯一識別通信方法，便可以建立連結，互相傳送資料，要使用 socket 的 kernel 結構須包含以下三個部分 ：

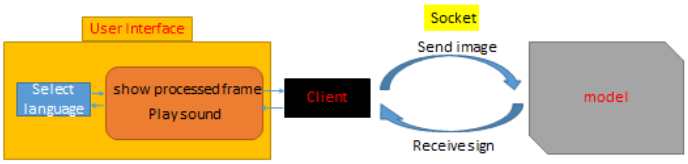
一. 插座層(socket layer)

二. 通訊協議層(protocol layer)

三. 設備層(device layer)

3. 系統架構與流程

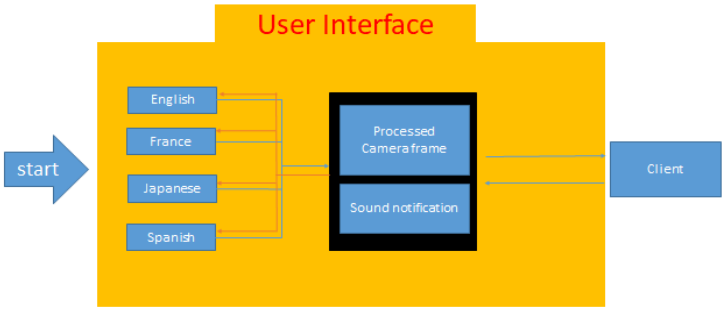
為了達到我們系統所需的要求，正確判斷路邊的交通號誌，更希望能在充足的反應時間內，用客製化的語音系統提醒駕駛者眼前出現的路標下圖是我們的整體系統架構



整體架構分為**使用者介面(手機端)**、**傳輸之實作**、以及神**經網路模型(電腦端)**這三項，以下將分述三個部分。

A.使用者介面(手機端)

在手機端的部分，我們對介面進行優化的排版，讓手機整體版面看起來乾淨舒適，另外在實作的部分，Opencv 幫助我們將手機鏡頭的畫面畫到螢幕上，另外所抓取到的每一禎 frame，原先是 bitmap 格式，我們轉換成 jpeg 格式，以利於之後將圖片傳輸到 server 神經網路模型上，另外，我們將翻譯語言分為四種，分別是英文、法文、西班牙文、以及日文，做法是將對應標誌語言放入 Android studio 中，當手機接收到號誌種類時，會將相對應的號誌名稱以及號誌圖像顯示於畫面左上角，最後聲音的部分，採用的是連續偵測到路標時才會做出聲音的回饋，我們將門檻(threshold)設定為第一次偵測到路標時要連續五次，而通過第一次時，之後的門檻值會上升到二十次，通常在快速的行車下，一次提醒已經足夠。



將每張 frame 的 byte 數目計算好，送至 server 中讓 server 開相同大小的空間，之後再將 frame 資訊送至 server 中，如此一來，便能達到即時接收，不延遲的圖片傳輸。

B. 傳輸之實作

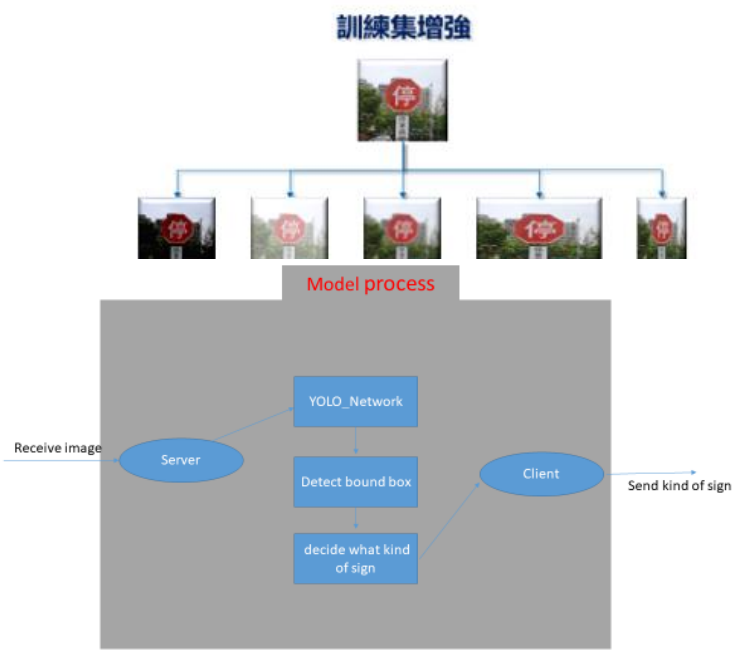
傳輸方面使用 socket 實作，理由是本專題要求即時性(real-time)，而 socket 以 byte 為底層的傳輸單位不僅使封包尺寸縮小，更能建構於 TCP 協定下的安全傳輸，在本專題中，要即時且立即的傳送手機鏡頭的每一禎 frame，故需要設計一方法使 server 端可以即時接收不延遲，方法為使用 Opencv 之功能擷取相機鏡頭的每一禎 frame，並且計算 frame 所佔的 byte 大小，而在相同尺寸的大小下為何 byte 數目會有差異呢？主要原因是在 jpeg 格式中，圖片會使用 RGB 的色彩模式進行編碼，故就算是在尺寸相同的情況下，每張 frame 也會跟隨環境亮暗程度不同而改變 RGB 的數字，進而使 byte 的數目不同，所以我們無法固定 server 的空間，來接收每禎 frame，所以我們只好先

C. 神經網路模型處理架構(server 端)

在 server 部分，我們搭載的是 YOLO 類神經網路為了要判斷交通號誌的種類，於是使用自定義的資料集，我們到台灣的道路上照取含有交通標誌的圖像，但為了簡化模型的大小與訓練時間，我們挑選出一寫含有中文字的交通標誌讓 YOLO 神經網路訓練，分別是

1. **停車再開**
2. **禁止臨時停車**
3. **左彎**
4. **禁止進入**
5. **禁止迴轉**
6. **當心行人**
7. **分道**
8. **禁止左轉**
9. **減速慢行**
10. **禁止停車**
11. **當心兒童**
12. **遵行方向(右)**

我們總共蒐集 1000 張左右的



訓練集，很明顯這對神經網路來說是不夠的，因此我們使用資料集增強的技術，對每張圖片做調高亮度、降低亮度、調整解析度、左右伸縮、上下伸縮的抗雜訊的運算，讓我們的資料集更加擬合現實道路情況，最後總共生成　12000 張訓練集(training set)，其中 500 張當作測試集(testing set)如下圖的圖示所示

而訓練完後的模型經過 500 張測試集的試驗，每一張測試集只含有一個交通標誌，最後成功辨識出 381 個交通標誌，準確率在 76.2%(Recall)左右，而在這 381 個交通標誌中，辨識正確的標誌數量是 381 個(Precision)，也就是說，只要有抓到交通號誌，那麼所辨識出的交通標誌一定是正確的，這邊要強調正確率是因為經過我們測試，這個辨識率已經足夠駕駛做出反應時間，那麼更重要的是判斷的標誌不是正確的，若我們能做到在相同的辨識率的情況之下，調高判斷正確的機率，我們的系統便可以做到即時性以及正確性。

完成模型的訓練後，便可以將手機傳輸來的 frame，放入模 型中進行判斷，而得出的結果在經由 socket 傳輸至手機端，整體模型程式結構如下

4. 系統結果與成品

起始畫面

選擇語言畫面

英文翻譯

西班牙文翻譯

日文翻譯

5. 評估與未來方向

就我們的動機與想法而言，我們實現了大部分的目標，目前成果還算滿意，而簡單的操作介面也符合使用者需求，整體而言，client-server 架構的速度也算令人接受，整體延遲時間不影響使用者之體驗，使用者僅需有網路便可以做到交通號誌的翻譯，因此功能方面有達到使用者需求。

再來是我們未來希望改進的方向，第一點，深度學習必然是未來的趨勢，未來深度學習的算法與模型必定更加準確以及輕量，物聯網以及深度學習結合是人類進步的方向，若是每個設備都可以將神經網路放入，那麼深度學習的應用一定可以更廣泛，像將 YOLO 整個放入手機中，並且準確度與速度都不降低，便可以做到無網路連線，也可以進行翻譯的系統，第二點，我們希望這個系統能夠推廣到外國的交通標誌，例如德國，荷蘭，韓國等台灣人較常去的國家，若有相對應的即時翻譯成中文的系統，便可以節省台灣人查找外國標誌的時間，以及在道路上立即反應時間，最後，本專題的未來創新性很高，我們希望將來有更多的想法應用在各不同層面上，如果未來有機會，我們也希望繼續朝目標發展改良。

6. 銘謝

在這一學期的專題中，我們特別感謝指導教授，教授從人工智慧的介紹，一路到指導我們專題的流程，這段期間幫了我們很大的忙，可以說我們有現在的成果都是由教授由零開始一手一手提攜而來的，教授耐心指導我們深度學習、神經網路、報告的邏輯過程，到最後的程式改進以及功能新增，這一年我們從懵懂無知的新手蛻變成深度學習領域的能手，當然我們的學習之路還很長，不過因為教授的指導縮短了許多距離，另外教授教會我們製作專題的流程，讓我們在未來做計劃甚至是研究所都有一個經驗，也很感謝教授在過程中不斷給予我們不同的意見，做出屬於我們最特別的專題系統。

7. 參考資料

[1] https://github.com/qqwweee/keras-yolo3

[2] https://grail.cs.washington.edu

/wpcontent/uploads/2016/09/
redmon2016yol.pdf

[3]
https://docs.python.org/3/how
to/sockets.html