國 立 虎 尾 科 技 大 學

電 子 工 程 系

專題製作報告

應用機器視覺技術多車種車牌辨識

安全管理系統

(License plate recognition system)

指導教授 :蔡振凱

班 級 :夜四電子四甲

製作學生 :曾品鈞、陳岳鋒、楊岡穎、蔡佳展

學號 :40540411、40540415、40540419、40540425

中華民國 108 年 11 月 5 日

[摘要 2](#_Toc28268308)

[**動機** 3](#_Toc28268309)

[**預期成果** 3](#_Toc28268310)

[**所需工具** 4](#_Toc28268311)

[**使用軟體介紹** 5](#_Toc28268312)

[**OpenCV** 5](#_Toc28268313)

[**應用領域** 5](#_Toc28268314)

[**Python** 6](#_Toc28268315)

[**歷史** 6](#_Toc28268316)

[**TensorFlow** 7](#_Toc28268317)

[解構 TensorFlow 7](#_Toc28268318)

[TensorFlow 如何融入 AI 與 Machine Learning 7](#_Toc28268319)

[Computational Graph 和概略理論 8](#_Toc28268320)

[**想使用 TensorFlow，開發者需要做什麼？** 8](#_Toc28268321)

[**TensorFlow 常見的三個應用方式** 9](#_Toc28268322)

[**Yolo：you only look once** 9](#_Toc28268323)

[**希望成果** 17](#_Toc28268324)

[理論 18](#_Toc28268325)

[**實際模式 Arduino 流程圖** 18](#_Toc28268326)

[**實際模式 Python 流程圖** 19](#_Toc28268327)

[**(一)** **硬體製作:** 21](#_Toc28268328)

[**(二)** **軟體撰寫:** 21](#_Toc28268329)

[結論 28](#_Toc28268330)

[建議 28](#_Toc28268331)

[參考資料 29](#_Toc28268332)

[操作手冊 30](#_Toc28268333)

# **摘要**

在現今生活中，汽機車已經是每個人不可或缺的交通工具，車輛的使用越來越頻繁，但也衍生出許多治安與控制問題，例如:交通違規、車輛遺失等情形。在事件發生之後，藉由車牌辨別車主已經是常態，並記錄進入車牌號碼及車型，可依過往進出紀錄判斷是否為附近居民，現今警政單位依然是依靠人工辨別，大幅增加員警工作量與精神消耗，為有效減少警力負擔，此車牌辨識可應用於辨識使用車輛停車場或管制區。

在資訊爆炸的時代，AI的興起勢不可檔，尤其是應用在視覺的AI更是大幅躍進，有別以往依靠正則法，只能辨識特定的物體，到使用類神經網路及各類演算法可以偵測各類物件，透過大量Data提高正確率，更可以自訂議物件類別，用途多樣且彈性，AI的興起，可以減緩人力的負擔，也不像人類有多餘的情感可達成判別正當性。

藉由arduino控制攝像頭傳輸至電腦做紀錄並判別是否為可進入車輛，程式使用Python搭配OpenCV作影像辨識，並加入最新的機器學習(Machine Learning)辨別車型種類，且建立一資料檔並做紀錄。卻認為可進入之車輛就給指令給arduino控制閘門打開。若為危險車輛並控制arduino通知管理員避免意外發生。

搭上5G列車，提前練習物聯網應用選用Arduino做為物聯網媒介，與電腦通訊，只需設定程式內碼即可控制多個攝影機，以學校來舉例，在排除攝相電纜長度，即在電算中心內部的虛擬電腦即可以控制校園所有出入口控制，並在有人誤闖校園時，第一時間通知警衛到場查看。僅僅需要一台電腦中央控制就能控制所有閘門。

## **動機**

每當我經過學校門口，時常看到有機車等待被警衛放行，他們並非不是校園教職員工，而單純只是校園車牌辨識系統僅能辨別小客車車牌，以致機車需要下車刷磁釦才能通過障礙，若能增進校園門口辨識系統想必須相當值得榮幸的事，並與專題組員討論過後，決定以此為題目作為大學專題內容。

現在最新型的停車場都是用車牌辨識系統來處理車輛，顯得更為重要，傳統用人工進行管理的方法不符合經濟成本於是才有車牌辨識系統的產生。還能記錄進出校門車輛紀錄，達到校園安全掌控。

## **預期成果**

傳統車牌辨識的做法有個缺點，就是在車牌位子有過多的裝飾容易造成機器判讀失敗，又或者未停留在規定區域及造成讀取失敗，且在判別汽機車上困難，不便於記錄，車牌讀取時，會因光影問題易造成讀取失敗(因為ocr辨識條件嚴苛)。

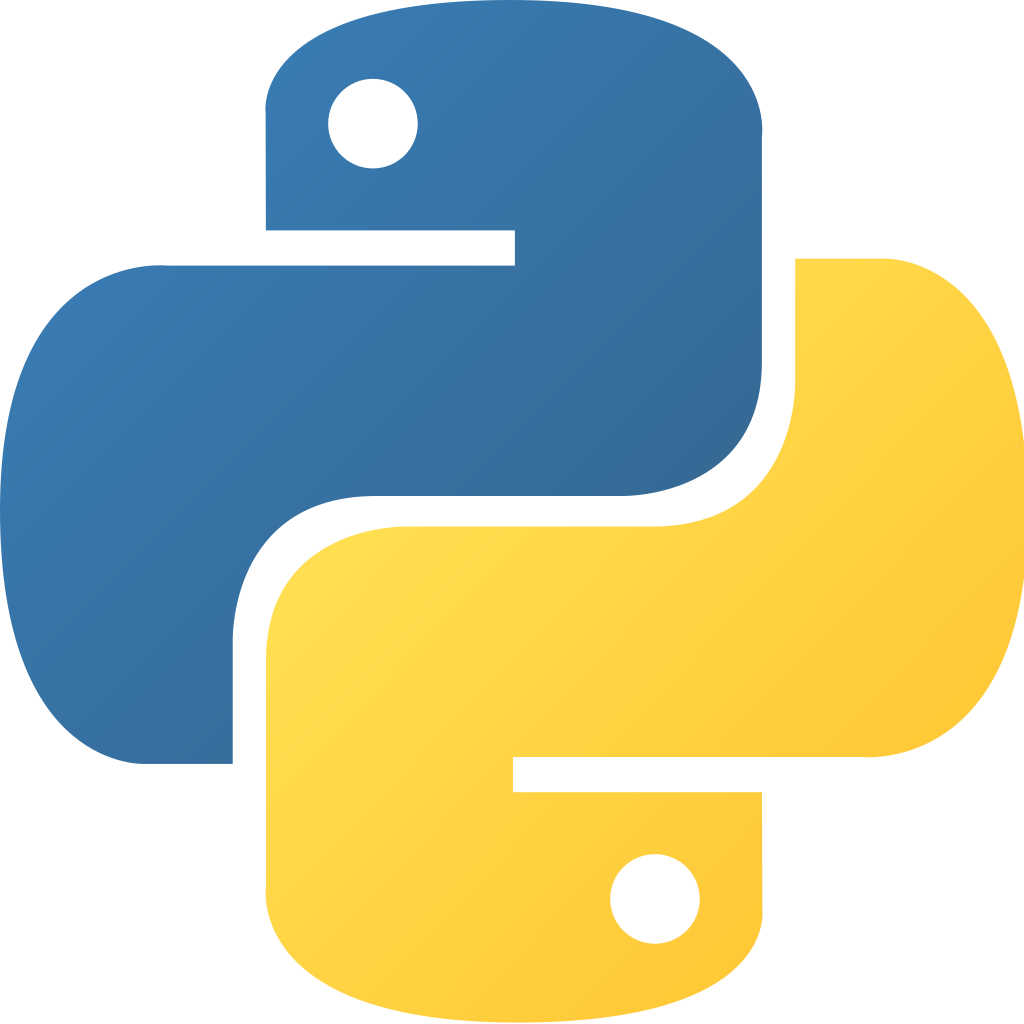
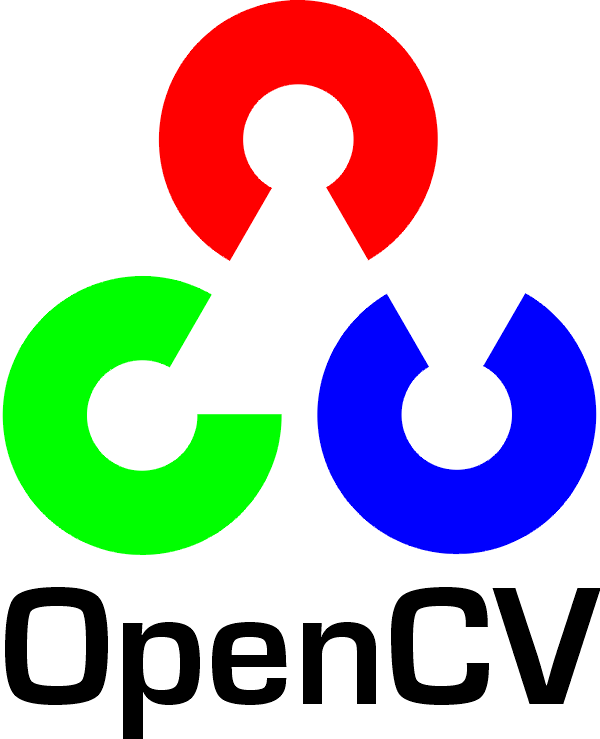
所以我們使用YOLO物件辨識(Object Detection)，學習如何在一張圖片中找出車種即車牌位子，並將它讀取出來，尋找到車牌位子使用圖片輪廓化，再將其作二值化，透過膨脹與收縮輪廓，可以找到線條區域密度較高的車牌位子，精確位子對於辨識更有幫助，找出車牌後利用訓練後的OCR辨識牌號碼並核對車號，是否異常，即通知停車場閘門開啟或者關閉。

當前較為流行的物聯網，我們使用Arduino控制閘門，即燈號控制，還有即時反應前方是否存在車輛，存在並發送請求核對是否為安全車牌。

若時間許可，想嘗試加入模板比對，或者CNN機器學習辨識車號，兩種分辨器相互比對，增加判別正確性，若車號誤判次數過多也不便日常使用。

**所需工具**

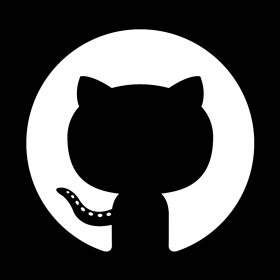
* 使用工具 : 羅技 C920r HD Pro 視訊攝影機、台達DC小型馬達、3D列印
* 電腦硬體 : AMD\_R7\_1700X、RAM\_16G、HDD\_500G、GPU\_GTX1070(總機)
* 開發程式 : 主要程式語言、主要開發套件OpenCV、主要程式編輯器Visual Studio Code、版本控制軟件Git、。

****

**羅技 C920r HD Pro 視訊攝影機**

**OpenCV**

**Python**



**GitHub**

## **使用軟體介紹**

**OpenCV**

　　OpenCV全名是Open Source Computer Vision Library，是一個影像處理函式庫，由Intel發起並參與開發，以BSD授權條款發行，可在商業和研究領域中免費使用，目前是非營利的基金組織OpenCV.org在維護。

　　OpenCV在影像處理方面應用廣泛，可以讀取儲存圖片、視訊、矩陣運算、統計、影像處理等，可用在物體追蹤、人臉辨識、傅立葉轉換、紋理分析、動態視訊的影像處理等。

　　OpenCV提供的函式方便我們推演更進階的影像處理演算法，就好像MATLAB的功用，但是執行速度比MATLAB快上許多，通常也比我們自己用C/C++寫的函式還快，而除了C/C++之外，OpenCV也提供其他語言的支援，像Java或Python等。

**應用領域**

　　影象資料的操作 ( 分配、釋放、複製、設定和轉換)。 影象是視訊的輸入輸出I/O （檔案與攝像頭的輸入、影象和視訊檔案輸出）。   
矩陣和向量的操作以及線性代數的演算法程式（矩陣積、解方程、特徵值以及奇異值等）。   
各種動態資料結構（列表、佇列、集合、樹、圖等）。   
基本的數字影象處理（濾波、邊緣檢測、角點檢測、取樣與差值、色彩轉換、形態操作、直方圖、影象金字塔等）。   
結構分析（連線部件、輪廓處理、距離變換、各自距計算、模板匹配、Hough變換、多邊形逼近、直線擬合、橢圓擬合、Delaunay 三角劃分等）。   
攝像頭定標（發現與跟蹤定標模式、定標、基本矩陣估計、齊次矩陣估計、立體對應）。   
運動分析（光流、運動分割、跟蹤）。   
目標識別（特徵法、隱馬爾可夫模型：HMM）。   
基本的GUI （影象與視訊顯示、鍵盤和滑鼠事件處理、滾動條）。

**Python**

　　Python 是物件導向程式級高階程式語言，也是直譯式程式語言。Python以強調對程式語言的語句易讀、易懂、易學(簡潔和清晰的語法特點)及加快程式開發的時效，方便使用，可以完成各種難度的應用，並可在大多數的系統中運行，以減少開發及維護成本的觀念進行發展。

　　Python 同時支援 modules 和 packages ，另外 Python 為跨平台程式語言也支援 unicode 字元。功能強大而完善的通用型語言，可以用於很多種軟體開發動態程式，使得 Python 非常有吸引力，發展至今已有十多年的歷史，成熟且穩定。

通常，有許多程式設計者接觸 Python 後都會被她的優點所吸引，並提供了許多自行開發的  library(函式庫)以提供其他 Python 程式設計者下載使用，所以具有豐富和龐大的類別函式庫，可以負荷支持平常大部份的應用，使得現今 Python 發展迅速及使用廣泛，相關資源也相當豐富。

**歷史**

　　Python 的創始人為 Guido van Rossum。在1989年聖誕節期間的阿姆斯特丹，Guido 為了打發聖誕節的無趣，決心開發一個新的腳本解釋程序，作為 ABC 語言的一種繼承。之所以選中 Python（大蟒蛇的意思）作為程序的名字，是因為他是一個 Monty Python 的飛行馬戲團的愛好者。可以說 Python 是從 ABC 發展起來，主要受到了Modula-3（另一種相當優美且強大的語言，為小型團體所設計的）的影響。並且結合了 Unix shell 和C的習慣。

　　早期 Python 是以 C 語言為基底撰寫而成，目前則有以 Java 語言為基底的 Jython 及架構在  .NET 的 IronPython 版本，甚至有以 Python 語言為基底的 PyPy 實作品，它可以將 Python 程式碼編譯成其他語言的執行檔，像是 C、CLI (.Net)、JavaScript 等。

　　Python 具有腳本語言中最豐富和强大的類庫，足以支持绝大多數日常應用。它的名字來源於一個喜劇，也許最初設計Python這種語言的人並没有想到今天 Python會在工業和科研上獲得如此廣泛的使用。

◎2019年10月30日從Dropbox退休，但並不代表Guido已經從Python退休

**TensorFlow**

Google 的 [TensorFlow](https://www.tensorflow.org/) 開源函式庫協助開發工程師建立計算圖（Computational Graph），內含免費的[附加模組](https://www.tensorflow.org/api_docs/python/tf/contrib)讓開發與消費品和移動應用程式（mobile apps）相關的人工智慧 / 機器學習軟體變得更容易。開發者需要具備 Python 或 C++ 程式語言基礎。本文將探討三個常用模組和延伸的商業發展潛力：

1. 卷積神經網路（Convolutional Neural Networks, CNN）適用於辨識圖像
2. 序列到序列（Sequence-to-Sequence, Seq2Seq) 模型適用於自然語言處理
3. 大型線性模型 (Linear Models)，適用於數據分析

解構 TensorFlow

自 2015 年 Google 開放原始碼，TensorFlow 即成為創建深度學習模型時使用的框架（Framework）。深度學習 （Deep Learning）是機器學習（Machine Learning, ML) 模型中會使用多層神經網路的其中之一。TensorFlow 函式庫讓使用者建立 computational graph 來套用不同功能。

TensorFlow 之所以席捲全球，除了因為是免費以外，主要因為它（相對）容易使用，即使 ML 初學者也可以接觸強大的函式庫，免於從零開始建立自己的 AI 模型。換句話說，TensorFlow 和附帶模組便利了 mobile apps 和後端服務的 ML / AI 軟體開發。

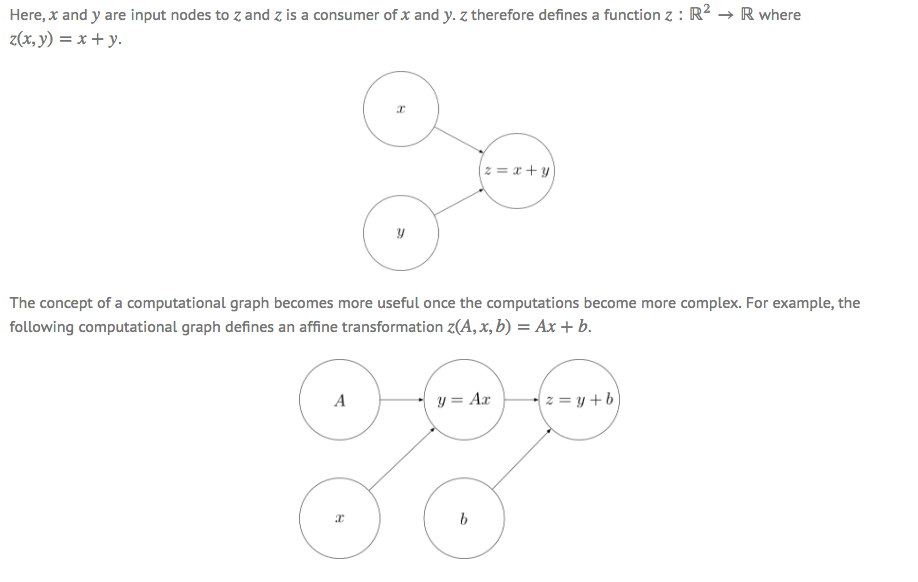
TensorFlow 如何融入 AI 與 Machine Learning

TensorFlow 是眾多 ML 函式庫裡之一（其他例子包括 [CNTK](https://github.com/Microsoft/cntk) 和 [Theano](https://github.com/Theano/Theano" \t "_blank)）。Machine Learning 屬於計算機科學的其中一個領域，在毋須直接編程的情況下讓[計算機](https://en.wikipedia.org/wiki/Computer)擁有學習能力（[註1](https://en.wikipedia.org/wiki/Machine_learning#cite_note-1)）。比方說，[AlphaGo Zero](https://deepmind.com/blog/alphago-zero-learning-scratch/) 的 AI 透過「自學」下圍棋，在對戰中完勝了曾擊敗人類世界冠軍的舊版 AlphaGo。Machine Learning 針對無法透過顯性演算法獲得最優解的任務尤其有用，比如用戶篩檢、有關高語境資料的排序、或用於預測和剖析的聚類分析 (Clustering)。

實際應用例子包括偵測舞弊或資料外洩、[電郵過濾](https://en.wikipedia.org/wiki/Email_filtering)、光學字元辨識 (Optical Character Recognition, OCR) 和排序等等。

下一節，我們將揭開 Computational Graph 的神秘面紗，幫助大家了解 TensorFlow 的關鍵和操作原理。

Computational Graph 和概略理論



想使用 TensorFlow，開發者需要做什麼？

TensorFlow 為發揮最佳效能，在特定的平台剔走了部分功能（可以參考 [TensorFlow Lite 和 TensorFlow Mobile](https://www.tensorflow.org/mobile/" \t "_blank)）， mobile 和 web 開發者可以利用函件庫為消費品打造 AI 支援的功能。對神經網路具備基本知識的工程師可以利用 framework 作資料集 (Data Set)、模擬器 (Estimator)、訓練 (Training) 和推論 (Inference) 等用途。

即使對神經網路沒有認識的開發者可能也會想要從高階神經網路 API 開始（譬如 [Keras.io](https://keras.io/)）。而以 Python 寫成的 Keras 可以在 [TensorFlow](https://github.com/tensorflow/tensorflow" \t "_blank)、[CNTK](https://github.com/Microsoft/cntk" \t "_blank) 和 [Theano](https://github.com/Theano/Theano" \t "_blank) 上運行，有利於簡單快速地製作原型 （Prototyping）。

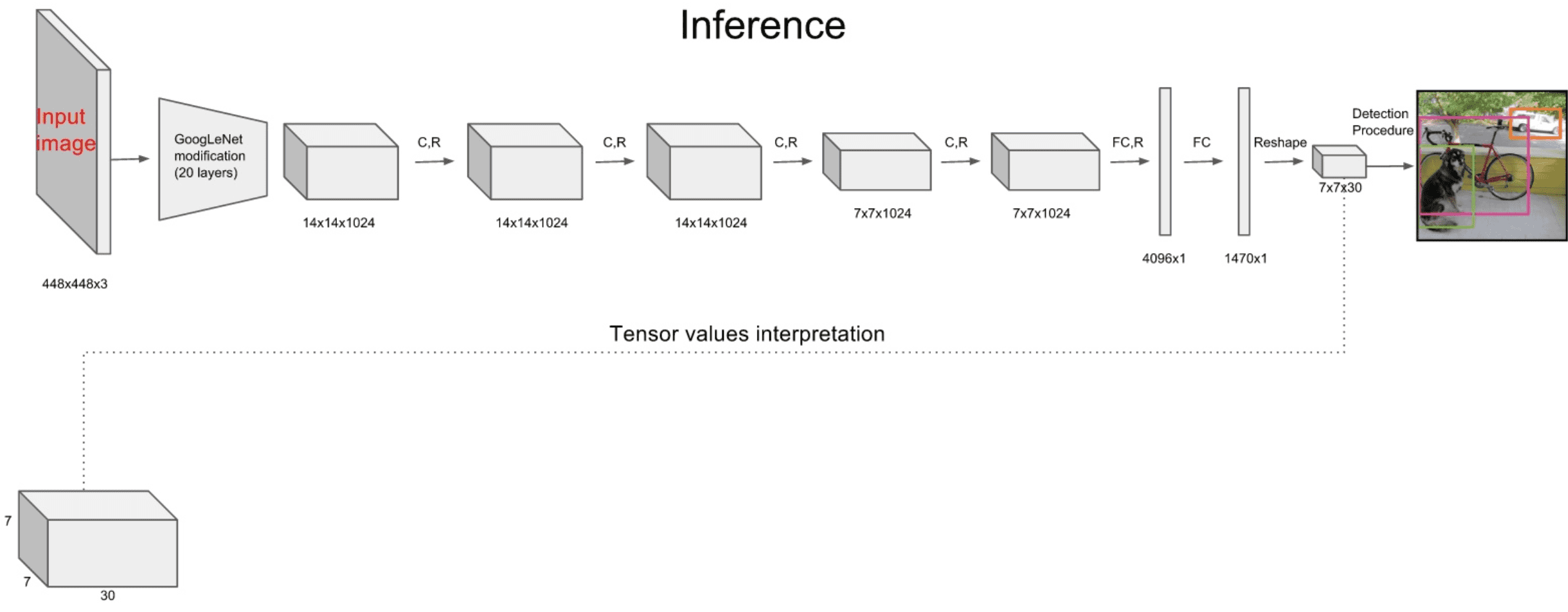
**要開始使用 TensorFlow ，開發工程師需要先具備以下的知識：**

1. Python 或 C++
2. 懂得如何使用 **Numpy的 array**（它是一個 Python 擴充的數值運算函式庫，TensorFlow 以此來處理陣列（Array）和矩陣（Matrix））
3. TensorFlow 常見的三個應用方式

[TensorFlow 函式庫裡包含的眾多模組](https://www.tensorflow.org/api_docs/python/tf/contrib)可以套用一系列 AI / ML 功能。以下三個是最廣泛被使用的模組：

1. 卷積神經網路（ Convolutional Neural Networks, CNN），適用於辨認和處理圖像
2. 序列到序列（Sequence-to-Sequence, Seq2Seq）模型，適用於處理自然語言相關的功能
3. 大型線性模型 (Linear Model)，適用於資料分析和簡單的行為預測

## **Yolo：you only look once**

針對於two-stage目標檢測算法普遍存在的運算速度慢的缺點，yolo創造性的提出了one-stage。也就是將物體分類和物體定位在一個步驟中完成。yolo直接在輸出層迴歸bounding box的位置和bounding box所屬類別，從而實現one-stage。通過這種方式，yolo可實現45幀每秒的運算速度，完全能滿足實時性要求（達到24幀每秒，人眼就認爲是連續的）。它的網絡結構如下圖

主要分爲三個部分：卷積層，目標檢測層，NMS篩選層

YOLO核心思想

YOLO核心思想就是利用整張圖做為輸入，直接在輸出層做bounding box的位子其所屬的類別。Faster-RCNN中也直接用整張圖做為輸入，但是faster-RCNN整體還是用proposal+classifier思想，只不過提取proposal步驟放在CNN中實現，而YOLO採直接回歸的思路。

YOLO實現方法

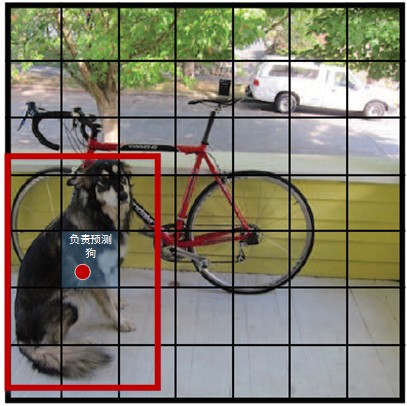
YOLO將輸入圖像分成S x S的格子，辨識物體的中心位子座標落入其中格子中，那這個格子就負責辨識出這個物體(圖一)。

每個格子預測X個bounding box及Confidence score，及c個類別概率。Bbox信息(x,y,z,h)為物體的中心相對格子位子的偏移及寬度和高度被歸一劃，置信度反應是否包含物體及包含物體的準確性，定義為Pr(Object)×IOUtruthpred,

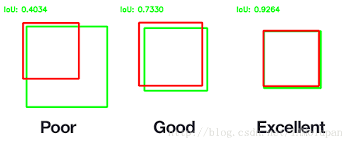
其中Pr(Object)∈{0,1}Pr(Object)×IOUpredtruth,其中Pr(Object)∈{0,1}.

這個 confidence 代表了所預測的 box 中含有 object 的置信度和這個 box 預測的有多準這兩重信息，其值是這樣計算的(圖二)：

示意圖:圖三



**圖二**



**圖一**

**圖三**

其中如果有 object 落在一個 grid cell 裏，第一項取 1，否則取 0。 第二項是預測的 bounding box 和實際的 groundtruth 之間的 IoU 值。

每個 bounding box 要預測 (x, y, w, h) 和 confidence 共5個值，每個網格還要預測一個類別信息，記爲 C 類。則 SxS個 網格，每個網格要預測 B 個 bounding box 還要預測 C 個 categories。輸出就是 S x S x (5\*B+C) 的一個 tensor。

注意：class 信息是針對每個網格的，confidence 信息是針對每個 bounding box 的。

舉例說明: 在 PASCAL VOC 中，圖像輸入爲 448x448，取 S=7，B=2，一共有20 個類別（C=20），則輸出就是 7x7x30 的一個 tensor。

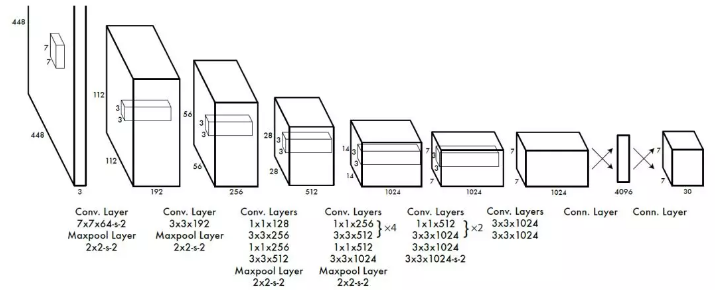
**YOLO網絡結構**

YOLOv1網絡借鑑了GoogLeNet分類網絡結構。不同的是，YOLO未使用inception module，而是使用1x1卷積層（此處1x1卷積層的存在是爲了跨通道信息整合）+3x3卷積層簡單替代。  
YOLOv1網絡在最後使用全連接層進行類別輸出，因此全連接層的輸出維度是 S×S×(B×5+C)S×S×(B×5+C)。  
YOLOv1網絡比VGG16快(浮點數少於VGG的1/3),準確率稍差。

缺餡:

1. 輸入尺寸固定：由於輸出層爲全連接層，因此在檢測時，YOLO訓練模型只支持與訓練圖像相同的輸入分辨率。其它分辨率需要縮放成改分辨率.

2. 佔比較小的目標檢測效果不好.雖然每個格子可以預測B個bounding box，但是最終只選擇只選擇IOU最高的bounding box作爲物體檢測輸出，即每個格子最多隻預測出一個物體。當物體佔畫面比例較小，如圖像中包含魚群或鳥群時，每個格子包含多個物體，但卻只能檢測出其中一個。

整個網絡結構如下圖所示(圖四)：

**圖四**

YOLO細節實現

每個 grid 有 30 維，這 30 維中，8 維是迴歸 box 的座標，2 維是 box的 confidence，還有 20 維是類別。 其中座標的 x, y 用對應網格的 offset 歸一化到 0-1 之間，w, h 用圖像的 width 和 height 歸一化到 0-1 之間。

在實現中，最主要的就是怎麼設計損失函數，讓這個三個方面得到很好的平衡。作者簡單粗暴的全部採用了 sum-squared error loss 來做這件事。

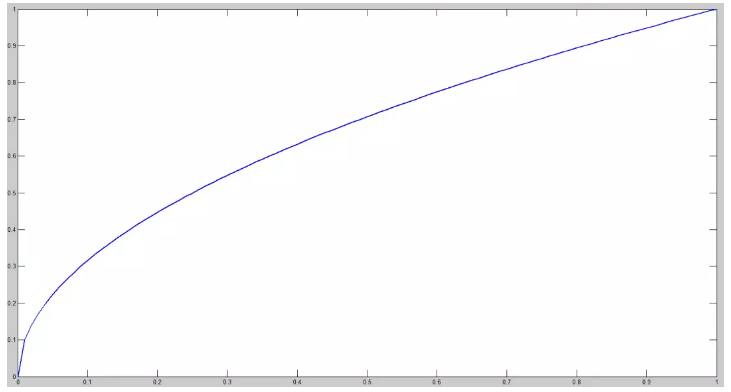
這種做法存在以下幾個問題：

第一:8維的 localization error 和20維的 classification error 同等重要顯然是不合理的；   
第二:如果一個網格中沒有 object（一幅圖中這種網格很多），那麼就會將這些網格中的 box 的 confidence push 到 0，相比於較少的有 object 的網格，這種做法是 overpowering 的，這會導致網絡不穩定甚至發散。

解決辦法：

1.更重視8維的座標預測，給這些損失前面賦予更大的 loss weight, 記爲在 pascal VOC 訓練中取 5。

1. 對沒有 object 的 box 的 confidence loss，賦予小的 loss weight，記爲在 pascal VOC 訓練中取 0.5。
2. 有 object 的 box 的 confidence loss 和類別的 loss 的 loss weight 正常取 1。
3. 對不同大小的 box 預測中，相比於大 box 預測偏一點，小 box 預測偏一點肯定更不能被忍受的。而 sum-square error loss 中對同樣的偏移 loss 是一樣。
4. 爲了緩和這個問題，作者用了一個比較取巧的辦法，就是將 box 的 width 和 height 取平方根代替原本的 height 和 width。這個參考下面的圖很容易理解，小box 的橫軸值較小，發生偏移時，反應到y軸上相比大 box 要大。（也是個近似逼近方式）



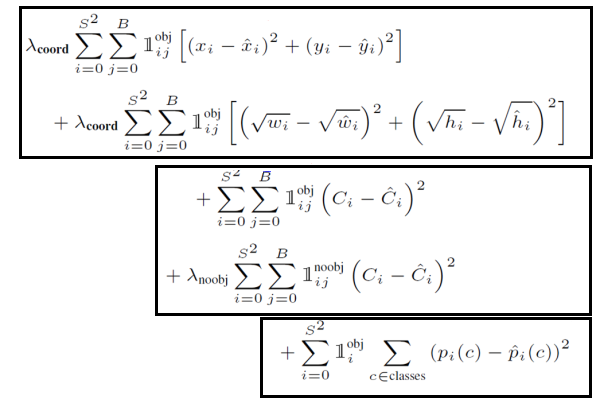
一個網格預測多個 box，希望的是每個 box predictor 專門負責預測某個 object。具體做法就是看當前預測的 box 與 ground truth box 中哪個 IoU 大，就負責哪個。這種做法稱作 box predictor 的 specialization。

**損失函數**

YOLO全部使用了均方和誤差作爲loss函數.由三部分組成:座標誤差、IOU誤差和分類誤差:

loss=∑i=0s2coordErr+iouErr+clsErrloss=∑i=0s2coordErr+iouErr+clsErr

簡單相加時還要考慮每種loss的貢獻率,YOLO給coordErr設置權重λcoord=5λcoord=5.在計算IOU誤差時，包含物體的格子與不包含物體的格子，二者的IOU誤差對網絡loss的貢獻值是不同的。若採用相同的權值，那麼不包含物體的格子的confidence值近似爲0，變相放大了包含物體的格子的confidence誤差在計算網絡參數梯度時的影響。爲解決這個問題，YOLO 使用λnoobj=0.5λnoobj=0.5修正iouErr。（此處的‘包含’是指存在一個物體，它的中心座標落入到格子內）。對於相等的誤差值，大物體誤差對檢測的影響應小於小物體誤差對檢測的影響。這是因爲，相同的位置偏差佔大物體的比例遠小於同等偏差佔小物體的比例。YOLO將物體大小的信息項（w和h）進行求平方根來改進這個問題，但並不能完全解決這個問題。

綜上，YOLO在訓練過程中Loss計算如下式所示：

**分類誤差**

**IOU誤差**

**座標誤差**

這個損失函數中：

只有當某個網格中有 object 的時候纔對 classification error 進行懲罰。

只有當某個 box predictor 對某個 ground truth box 負責的時候，纔會對 box 的 coordinate error 進行懲罰，而對哪個 ground truth box 負責就看其預測值和 ground truth box 的 IoU 是不是在那個 cell 的所有 box 中最大。

YOLO缺點

\* YOLO 方法模型訓練依賴於物體識別標註數據，因此，對於非常規的物體形狀或比例，YOLO 的檢測效果並不理想。

\* YOLO 採用了多個下采樣層，網絡學到的物體特徵並不精細，因此也會影響檢測效果。

\* YOLO 的損失函數中，大物體 IOU 誤差和小物體 IOU 誤差對網絡訓練中 loss 貢獻值接近（雖然採用求平方根方式，但沒有根本解決問題）。因此，對於小物體，小的 IOU 誤差也會對網絡優化過程造成很大的影響，從而降低了物體檢測的定位準確性。

YOLO 對相互靠的很近的物體，還有很小的群體檢測效果不好，這是因爲一個網格中只預測了兩個框，並且只屬於一類。

同一類物體出現的新的不常見的長寬比和其他情況時，泛化能力偏弱。

由於損失函數的問題，定位誤差是影響檢測效果的主要原因。尤其是大小物體的處理上，還有待改善。

則我們使用是YOLOv3 結構更為複雜，都可以通過改變模型結構提高準確度與精度。

**總結YOLO優缺點:**

快速(圖五)，在608 \* 608 解析度下分辨率可達20 FPS 。學習較為簡單。

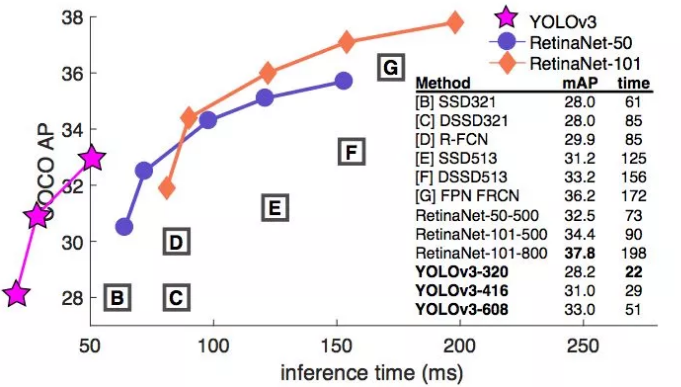
背景誤檢率低。

通用性強。YOLO對於藝術類作品中的物體檢測同樣適用。它對非自然圖像物體的檢測率遠遠高於DPM和RCNN系列檢測方法。

但跟RCNN系列物體檢測方法，YOLO具有以下明顯缺點：

識別物體位置精準性較差。

召回率低。在每個網格中預測固定數量的bbox這種約束方式減少了候選框的數量。



**圖五**

**研究目的**

在許多公眾停車場都可以看見車牌辨識，但因為現今通用的車牌辨識還是老舊的演算法處理，僅能辨識車牌與車牌號碼，不能辨識車種，導致機車與汽車的停車場只能分開停放，由於機車輪廓線條與角度比汽車多樣，導致機車辨識錯誤率大幅提高，所以多數的車牌辨識只能運用在汽車上面，所以我們使用最新技術AI演算法，使用AI幫我們尋找車牌，為了增加速度使用OCR技術辨識其中文字，希望結合現代技術，就可以實現汽機車都可以使用的車牌辨識技術，達到完整利用空間和彈性使用的效果。

**希望成果**

希望成果，我們以校園作為範本，預先設立安全車牌(可通行車輛)，當車輛觸碰到觸發點由Arduino回傳呼叫電腦啟動攝影相機進行拍照截圖，並用YOLO進行物件辨識，尋找車牌與辨識車種(可透過訓練改變想要的辨識的車型)，擷取車牌部分進行OCR辨識車碼，與安全車牌進行比對，過程以黃燈表示;比對成功回傳呼叫Arduino做打開閘門的動作並亮起綠燈意識通行;比對失敗即亮紅燈，表示非安全車牌需通知執勤人員，達到校園安全。

若時間允許情況下可加入GUI介面更貼近人意，放便人為操控，並且可加入複數辨識方法，如CNN掃圖辨識，切自模板辨識等，避免單單依靠OCR辨識，如果可以加入複數辨識方法，就能足以應付更多場景變化及光影問題。

**理論**

**實際模式 Arduino 流程圖**

**開始**

NO

Arduino 檢測觸點

驅動視訊鏡頭

Arduino回傳電腦

執行YOLO程式

YES

關閉紅色LED

打開黃色LED

檢測回收資料

安全車牌

非安全車牌

啟動閘門

亮起綠色LED

亮起紅色LED

**實際模式 Python 流程圖**

**開始**

**實驗步驟**

NO

接受到Arduino訊息

YES

執行YOLO程式

物件偵測

車型 車牌 號碼顯示

安全車牌

視訊截圖

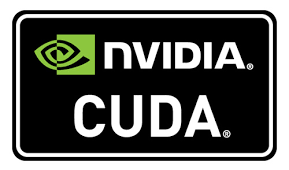
傳送是否為安全車牌代碼

首先我們必須先訓練好我們所想辨識的物件，如我們是辨識「機車」、「小客車」、「車牌」，所以需要訓練所需要辨識的權重檔案，因為YOLO能接受的檔案為YOLO格式，並不方便直接編輯使用所以我們利用Labelimg程式(圖一)框選標記物件成XML，再利用轉譯程式(make\_main\_txt.py)成YOLO檔案。



圖二

圖一

圖五

圖四

圖三

框選好照片，需使用裡面的分類程式(voc\_annotation.py)，將訓練檔分為訓練集、驗證集、測試集，在訓練前必須先滿足訓練環境，如Anaconda(圖二)、TensorFlow(圖三)、Keras(圖四)等等程式，修正train.py訓練程式中的路徑及可開始訓練，為了增加訓練速度我們有使用NVIDAI GPU用CODA(圖五)加速增加訓練速度。

訓練完成後(trained\_weights\_final.h5)，修正YOLO使用的權重路徑，即可開始測試訓練效果。(可參考網站教學<https://reurl.cc/6gRy5M>)

主程式寫在yolo.py含arduino 通訊阜，若要直接實行，必須修改部分程式碼通訊阜部分，也因訓練出的模型效果不同，需要修改一些參數在本書後面程式碼介紹會做詳細的解釋。

開始編輯程式，達到想要的成果。

1. **硬體製作:**

停車場控制本篇選用Arduino ESP-8266 Deving

控制伺服馬達開關停車場閘門，Button感應車體進入與離開，使用三種不同的發光二極體LED做停車場指示燈。

1. **軟體撰寫:**

修改訓練類別class (voc\_annotation.py)

1. **import** xml.etree.ElementTree as ET
2. **from** os **import** getcwd
4. sets=[('2007', 'train'), ('2007', 'val'), ('2007', 'test')]
6. classes = ["car\_card", "car", "moto"]

以及model\_data\my\_class.txt同訓練檔物件class

載入我們需要函數庫

1. **import** csv #載入csv模組
2. **import** pytesseract # 載入ocr模組
3. **from** PIL **import** Image #載入image 的 pil含數
4. **import** cv2 # 載入opencv
5. **import** serial # 載入 通訊arduino模型
6. **import** numpy as np # 載入可協助資料處理幫助圖片解析

將我們在過程中所會用到的全域變數先將其設定

1. COM\_PORT = 'COM3'    # 指定通訊埠名稱
2. BAUD\_RATES = 115200    # 設定傳輸速率
3. ser = serial.Serial(COM\_PORT, BAUD\_RATES)   # 初始化序列通訊埠
4. label\_test = [] #做為車種存取
5. Safenumber=["MPT3983","165BFE","BDC0771","2251WJ"] # 安全車牌設立
6. code = "" # 做為資料處理

我將部分程式打包成模塊，就先由模塊介紹

Def test是做yolo辨識 並 分割車牌圖片

1. **def** test\_new():
2. **try**:
3. image = Image.open(path\_0)    # 載入路徑圖片
4. **except**:
5. **print**('Open Error! Try again!')
6. **else**:
7. r\_image= yolo.detect\_image(image) # 使用yolo辨識路徑圖片
9. r\_image.show() # 顯示圖片
11. # yolo.close\_session()

14. b = [v.split(' ')[0] **for** v **in** label\_test] # 解析label文字串
15. b = [''.join(b)]
17. car1 = "carc"
18. moto1 = "moto"
19. dcar1 = "dcar"
21. **for** i **in** b:
22. **if** i[0:4]  == moto1 :
23. **print**("車種:機車")
24. **break**
25. **elif** i[-0:-4] == moto1 :
26. **print**("車種:機車")
27. **break**
28. **elif** i[0:4] == car1 :
29. **print**("車種:小客車")
30. **elif** i[-0:-4] == dcar1 :
31. **print**('車種:小客車')
32. **else**:
33. **pass**

36. # 讀取先前存取的座標路徑
37. with open("car\_test\_1.csv",'r',encoding='UTF-8') as cartest\_open:  #newline='' 是為了讓換行更可以被解析
38. rows = csv.reader(cartest\_open)
40. **for** row **in** rows:
42. row\_test = row[0]
43. # x,y,w,h 座標分別為 row[1-4] 以下做為解析列表
44. OK\_X = int(row[1]) -int(0)
45. OK\_y = int(row[2]) -int(0)
46. OK\_w = int(row[3]) +int(0)
47. OK\_h = int(row[4]) +int(0)
49. path\_1 = cv2.imread("fps.jpg")
50. # 切割row列表所提供的座標
51. ok\_image = path\_1[OK\_y:OK\_h, OK\_X:OK\_w]
52. cv2.imwrite("rgb\_card.png",ok\_image)
53. # 以下是存一張二值圖 稍後可以做程ocr辨識
54. Gray\_ok\_image = cv2.cvtColor(ok\_image,cv2.COLOR\_BGR2GRAY) #讀取鏡頭畫面並灰化
56. ret,thresh\_car\_card = cv2.threshold(Gray\_ok\_image,100,255,cv2.THRESH\_BINARY) #車牌切割後　二值化
58. cv2.imwrite('car\_card.png', thresh\_car\_card)   #輸出切割後車牌
60. cap.release()

因為我們所訓練出的物件辨識只能抓致大概位子其標示不夠精準，所以我們需要掛一個濾波器，做法就像傳統的車牌辨識。

1. **def** detect(img):
2. gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)
3. # 高斯平滑
4. gaussian = cv2.GaussianBlur(gray, (3, 3), 0, 0, cv2.BORDER\_DEFAULT)
5. # 中值濾波
6. median = cv2.medianBlur(gaussian, 5)
7. # Sobel算子，X方向求梯度
8. sobel = cv2.Sobel(median, cv2.CV\_8U, 1, 0, ksize = 3)
9. # 二值化
10. ret, binary = cv2.threshold(sobel, 140, 255, cv2.THRESH\_BINARY)
11. # 膨脹和侵蝕操作的核函数
12. element1 = cv2.getStructuringElement(cv2.MORPH\_RECT, (9, 1))
13. element2 = cv2.getStructuringElement(cv2.MORPH\_RECT, (7, 6))
15. # 膨胀一次，讓輪廓突出
16. dilation = cv2.dilate(binary, element2, iterations = 1)
17. # 侵蝕 去除細節
18. erosion = cv2.erode(dilation, element1, iterations = 1)
19. # 再次膨胀，凸顯輪廓
20. dilation2 = cv2.dilate(erosion, element2,iterations = 1)
21. # print(dilation2)
22. # cv2.imshow('dilation2',dilation2)
23. cv2.imwrite("dilation2.jpg",dilation2)
24. cv2.waitKey(0)
25. # return dilation2
26. a = np.array(dilation2) # 通過矩陣讀取二質化圖片
27. b = np.where(a==255) # 尋找白色像素點
29. test=[]
30. test.append(b)
31. # print(b)
32. card\_xy=[]
33. **for** i **in** test:
34. **for** a **in** i: #尋找dilation2最大最小白色像素點
35. x\_= min(a)
36. y\_= min(a)
37. w\_= max(a)
38. h\_= max(a)
40. card\_xy.append(x\_)
41. card\_xy.append(y\_)
42. card\_xy.append(w\_)
43. card\_xy.append(h\_)
44. #下面這行是切出非空白區域
45. img\_test = img[card\_xy[0]:card\_xy[2],card\_xy[4]:card\_xy[6]]
47. gray\_img\_test = cv2.cvtColor(img\_test, cv2.COLOR\_BGR2GRAY)
48. cv2.imwrite("img\_test.jpg",gray\_img\_test)

切出較好的車牌後，就可以進行OCR辨識了，我們使用比較簡易的OCR軟體(Tesseract-OCR)，引用OCR套件使用OCR辨識。

1. **def** OCR\_test(Image\_card):
2. pytesseract.pytesseract.tesseract\_cmd = 'C:\Program Files (x86)\Tesseract-OCR/tesseract.exe'
4. Aimage = cv2.imread(Image\_card)
5. code = pytesseract.image\_to\_string(Aimage)
6. car\_card\_number = pytesseract.image\_to\_string(Aimage)
8. code = code.replace(" ","");
9. code = code.replace("-","");
10. **return** code

上面介紹完包裝過後的模塊，以下是我們程式主體。

此外我們有額外做個比對，做為相互比對提高成功率，但因時間不足做好模板所以功能並不齊全，餘力寫至切出車牌並自母切割。

1. **def** segmentation(card\_segmentation):
2. # 1、讀取圖像，把圖向轉為灰階
3. img = cv2.imread(card\_segmentation)  # 讀取圖片
4. img\_gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)   # 灰階
5. # cv2.imshow('gray', img\_gray)  # 顯示
6. cv2.waitKey(0)
8. # 2、將灰階圖二值化，阈值是100
9. img\_thre = img\_gray
10. cv2.threshold(img\_gray, 100, 255, cv2.THRESH\_BINARY\_INV, img\_thre)
11. # cv2.imshow('threshold', img\_thre)
12. cv2.waitKey(0)
14. # 3、保存黑白圖片
15. cv2.imwrite('thre\_res.png', img\_thre)
17. # 4、分割字符
18. white = []  # 紀錄每一列的白色像素和
19. black = []  # ..........黑色.......
20. height = img\_thre.shape[0]
21. width = img\_thre.shape[1]
22. white\_max = 0
23. black\_max = 0
24. # 計算每一列的黑白色像素和
25. **for** i **in** range(width):
26. s = 0  # 這一列白色總数
27. t = 0  # 這一列黑色總数
28. **for** j **in** range(height):
29. **if** img\_thre[j][i] == 255:
30. s += 1
31. **if** img\_thre[j][i] == 0:
32. t += 1
33. white\_max = max(white\_max, s)
34. black\_max = max(black\_max, t)
35. white.append(s)
36. black.append(t)
37. # print(s)
38. # print(t)
40. arg = False  # False表示白底黑字；True表示黑底白字
41. **if** black\_max > white\_max:
42. arg = True
44. # 分割圖像
45. **def** find\_end(start\_):
46. end\_ = start\_+1
47. **for** m **in** range(start\_+1, width-1):
48. **if** (black[m] **if** arg **else** white[m]) > (0.85 \* black\_max **if** arg **else** 0.85 \* white\_max):  # 0.95這個参數可以調整，對應下面的0.05 總和為1
49. end\_ = m
50. **break**
51. **return** end\_
53. n = 1
54. start = 1
55. end = 2
56. **while** n < width-2:
57. n += 1
58. **if** (white[n] **if** arg **else** black[n]) > (0.15 \* white\_max **if** arg **else** 0.15 \* black\_max):
59. # 用来判斷是白底黑字還是黑底白字
60. # 0.05参数可调整，須對應上面的0.95
61. start = n
62. end = find\_end(start)
63. n = end
64. **if** end-start > 5:
65. cj = img\_thre[1:height, start:end]
66. # cv2.imshow('caijian', cj)
67. # cv2.imwrite("1.png",cj)
69. cv2.waitKey(0)

第23行模塊(segmentation)，和21行(detect)功能相同，但依舊保留裡面含分割車牌日後可以做模板比較。

1. **if** \_\_name\_\_ == '\_\_main\_\_':
3. # 打開yolo辨視 fps.jpg
4. yolo=YOLO()
5. **try**:
6. **while** True:
7. data = ser.readline().decode("utf-8")
8. **if** "\n" **in** data:
9. cap = cv2.VideoCapture(0) #設定 攝影機標籤
10. ret,camer\_frame = cap.read() # 讀取攝影機
12. cv2.imwrite('fps.jpg',camer\_frame) # 寫入圖片
13. cap.release() #釋放攝影機
15. path\_0 = r'fps.jpg'
16. test\_new() #執行 def test\_new()
17. label\_test = [] # 清空車種類別label 避免重覆執行 資料累加
19. imagePath = 'rgb\_card.png'
20. img = cv2.imread(imagePath)
21. detect(img)
22. card\_segmentation = "img\_test.jpg"
23. segmentation(card\_segmentation)
24. Image\_card = "thre\_res.png"
25. code = OCR\_test(Image\_card)
26. **print**("車牌號碼:",code)
27. **for** i **in** Safenumber:
28. # print(i)
30. # print(type(i))
31. **if** i == code:
32. **print**("安全車牌 開啟閘門")
33. ser.write(b"Servo\_ON\n")
34. **break**
35. **else**:
36. # pass
37. # print("Flase")
38. ser.write(b"Servo\_OFF\n")
39. cv2.destroyAllWindows()
40. cv2.waitKey(0)



45. **except** KeyboardInterrupt:
46. ser.close()    # 清除序列通訊物件
47. cv2.destroyAllWindows()
48. **print**('再見！')

**結論**

**建議**

**參考資料**

程式前言

<https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/541606/>

阿州的程式教學

<http://monkeycoding.com/?p=514>

宅學習

<https://sls.weco.net/CollectiveNote20/Python/Intro>

Pchome

<https://shopping.pchome.com.tw/>

git

<https://github.com/>

<https://backlog.com/git-tutorial/tw/>

yolo

<https://mc.ai/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ml-noteyolo%E5%88%A9%E7%94%A8%E5%BD%B1%E5%83%8F%E8%BE%A8%E8%AD%98%E5%81%9A%E7%89%A9%E4%BB%B6%E5%81%B5%E6%B8%ACobject-detection%E7%9A%84%E6%8A%80%E8%A1%93/>

<https://medium.com/@chenchoulo/yolo-%E4%BB%8B%E7%B4%B9-4307e79524fe>

<https://www.itread01.com/content/1544112970.html>

python

<https://www.python.org/downloads/>

OCR光學辨識

http://www.g4.com.tw/index.php

https://kknews.cc /tech/395rm33.html

http://www.jsocr.com/syjc/cjwt/156.html

https://kknews.cc/zh-tw/tech/6knlrqm.html

https://www.itread01.com/content/1544289877.html

**操作手冊**