國立中興大學電機工程學系研究所

碩士學位論文

基於建模軟體取得室內空間尺度與影像資訊以建立深度學習室內定位/地圖繪製之資料庫

Indoor Spatial and Image Information Inquiry through a 3D Modeler for a Deep Learning Indoor Positioning/Mapping Database

指導教授：林維亮 Dr. Wei-Liang Lin

研究生：陳漢庭 Han-Ting Chen

中華民國 一百零七年十二月

國立中興大學電機工程學系

碩士學位論文

題目(中文)： 基於建模軟體取得室內空間尺度與影像資訊以建立深度學習室內定位/地圖繪製之資料庫

題目(英文)： Indoor Spatial and Image Information Inquiry through a 3D Modeler for a Deep Learning Indoor Positioning/Mapping Database

姓名： 陳漢庭　　學號：7105064331

經　口　試　通　過　特　此　證　明

論文指導教授 ____林維亮____

論文考試委員 ____蔡清元____

____林惠勇____

____林維亮____

國立中興大學
電機工程學系

中　華　民　國　107　年　12　月　11　日

誌謝

首先真的感恩我的同學們，隆揚及至辰，如果今年4月隆揚沒有提醒我的話，我可能沒有辦法意識到，自己對於研究所的課業那麼不上心。不敢說本論文對於實驗室有多大貢獻，但我們的研究，著實因為我們的共心團結而完成了。感恩至辰在即將畢業前幫助我完成了許多畢業相關的流程，如果沒有隆揚、至辰以及學弟們的幫助，我可能沒有辦法那麼順利的完成論文。

剛開始來到研究所，從未想過能夠和其他實驗室感情那麼融洽，感恩能夠認識蔡曉萍老師、蔡清池老師、張振豪老師、林泓均老師、賴永康老師、江衍忠老師、吳崇賓老師、溫志煜的研究生們，以及系辦悅真姊、翁翁、張助教、林助教等的陪伴，讓我真的覺得來到研究所真的無比幸福。

我相信很多人上了研究所，都會經歷一些挫折和迷惘，例如讀了一段時間，會問自己：「為什麼我要讀研究所?」，或者與教授、同學相處上面遇到一些障礙。在研究上及論文上，我都起過負面意識，但是從現在的時間點往回看，過去所有的經歷都是有意義的。真的很感恩維亮老師的教導，如果沒有遇上老師，沒有辦法經歷這一切。

摘要

為了改善因不熟建築物內部結構而在室內迷路的問題，本團隊研究利用手機來做室內定位與繪製平面圖。其深度學習演算法，需要大量不同房間的室內資訊。因取得現實世界的資料不易，故本論文以 SketchUp 輸出的虛擬資料作為輔助。本論文著重於如何使用 SketchUp 和 Python 自動化提供虛擬資料。

本論文也研究虛擬貓狗與真實貓狗資料集，並得到虛擬貓狗資料能夠輔助真實貓狗資料的結果。

Abstract

To solve the problem of getting lost indoors of an unfamiliar building, Our team proposes photos taken by a mobile phone and its inertia information to position and sketch the indoor floor plan. The deep learning algorithm in the research of this team requires a lot of indoor information in different rooms. It is not easy to obtain real-world data.Therefore, this paper uses the virtual data output by SketchUp as an aid. This paper focuses on providing virtual data by using SketchUp and Python automation.

This thesis also trains a deep learning algorithm by different data sets (virtual cat/dog and real cat/dog). The result shows Transfer Learning can improve the accuracy.

目錄

圖目錄

表目錄

第一章 緒論

1.1 研究動機

為了改善因不熟建築物內部結構，而在室內迷路的問題，本團隊研究目標為「室內定位與製圖」，是利用手機拍出來的照片辨別出該空間的室內結構，以改善在室內迷路的狀況。

圖 1. Google 室內地圖示意圖

我們的願景是希望能夠在 Google 的 GPS 系統內加入室內地圖。以圖 1 為例，圖 1 的紅色框為 Google 地圖的室內區域，圖 1 內中間紅色箭頭指向室內空間建築物外觀，右邊紅色箭頭指向室內地圖。預想的情境為使用者進入某間超商，便可知曉該超商的室內地圖，以及使用者的所在位置。推廣此想法到結構更複雜的百貨公司，使用者可以輕鬆的透過手機，了解百貨公司的空間分布，以及使用者位置。

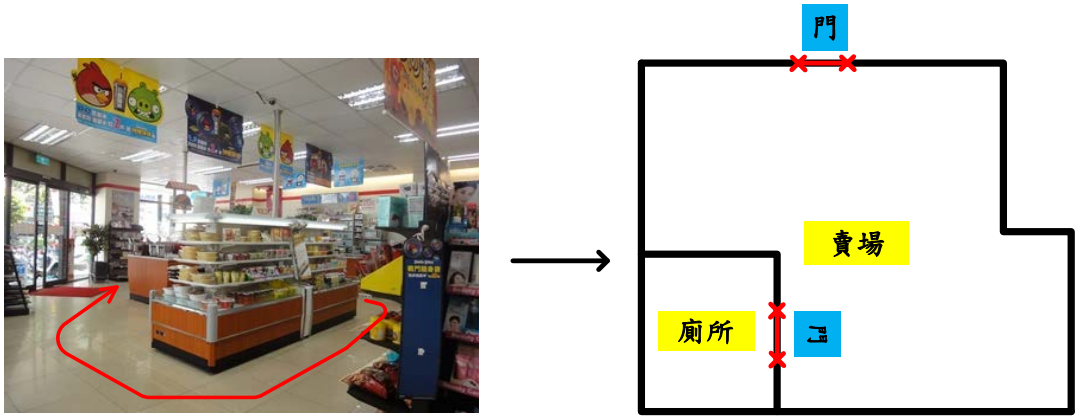


圖 2. 室內地圖建立示意圖

當大家都在使用 Google 室內地圖，在 Google 的雲端資料庫便有大量資料。儘管不同的使用者提供帶有雜訊的資料，造成系統預測同一個地點有不同預測結果，但當大家都用同一個資料庫，會使資料庫的室內地圖，在不斷更正下，趨向正確。使用者在行進間同時錄影，影像資訊搭配手機內建的慣性系統，即可辨識出房間的結構，並畫出平面圖，如圖 2 所示。

團隊的研究是利用圖片或相機拍下來的照片抽取資料，在能夠辨識前，深度學習演算法需要透過大量的訓練，讓電腦能夠在判斷階段時精準的畫出地圖。若我們只利用真實的資料訓練，因真實的資料需要大量的測量跟現場攝影，然而我們沒有大量的人力及時間建立大量真實資料，所以不可行。

由於真實世界的資料難以建立，我們需要利用虛擬資料的輔助來幫忙資料的多樣化。利用 3D 建模軟體來建造大量的虛擬環境，在虛擬環境中，我們可以透過自動化創造更多樣性的虛擬資料，再利用虛擬資料輔助真實的資料。經過評估，最後決定使用 SketchUp 進行虛擬室內環境採樣。

1.2 論文大綱

本篇論文共分為五章。從第二章開始，本論文先藉由貓與狗的二元辨識導入本團隊研究方向。第三章開始詳細介紹如何將從網路上獲取模型以建立數據集。第四章為本論文的實驗結果。最後，第五章為結論與未來展望。

第二章 深度學習數據集探討

此章節中，我們介紹與本論文相關研究。在 2.1 節，我們介紹在自駕車領域中，大家如何建立虛擬資料，及將真實與虛擬資料進行整合。而在 2.2 節，我們利用貓與狗的二元辨識， 驗證虛擬資料的可行性。

2.1 相關文獻

自駕車是最近機器人學領域中常常被大家討論的題材，而近年來，人工智慧也逐漸踏入自駕車的領域。而除了機器學習本身的架構外，訓練資料來源也是一個重要的議題，而這也正是本論文所關注的焦點。

在[8][9][10]中提到，機器學習的障礙來自於大量的資料需要大量的人工標註，在自駕車的領域中，資料的人工標註除了過於耗時之外，許多的情況無法被考慮進去，例如：前方橋斷掉或路失火等極端情況。除此之外，用於自駕車的資料必須考慮更周全，像是早上下雨、中午下雨、晚上下雨、濃霧等狀況，也都是在自駕車領域中需要考慮的問題。

為了解決人力標註過於耗時等相關問題，[8][9][10]提到虛擬資料以及電腦的自動標註方法。其所提到的訓練資料，來自遊戲引擎中的場景。利用遊戲引擎的 API，取得遊戲的場景及所需資料，並利用其他程式標註資料。圖 3 是以遊戲引擎場景為例，這些場景會因為白天或晚上，以及複雜的天氣變化，呈現各種不同，卻又逼真的渲染效果。這些場景幾乎可讓人誤以為現實世界的場景，故可以當作自駕車的訓練資料。

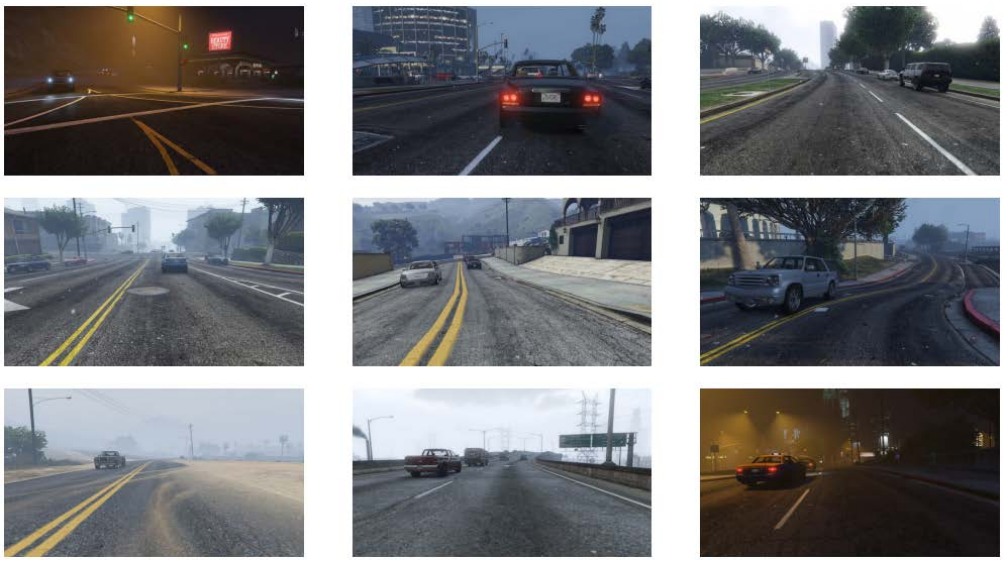


圖 3. 遊戲引擎場景[10]

自駕車的訓練及驗證上有許多方式，有些是利用遊戲引擎的資料直接訓練機器，而在驗證階段直接用真實的場景預測；有些在訓練階段時將遊戲引擎的虛擬資料及真實場景混在一起，在圖 4 的例子中，[10]的作者，先利用物件偵測，再利用邊緣偵測的方式讓資料只留下所需物體邊緣特徵，而預測階段時將現實的場景先邊緣偵測，再進行預測。

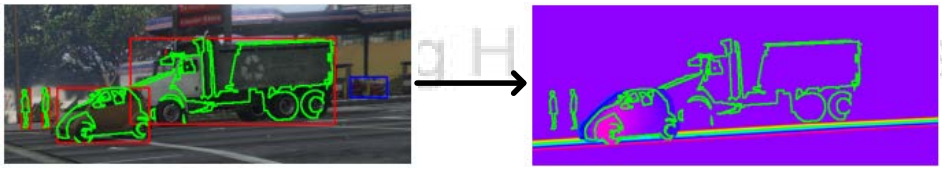


圖 4. 邊緣偵測[10]

在[10]中提到，虛擬資料必須具備能夠讓電腦認知為真實的場景，如此一來，這些虛擬資料才能成功輔助真實的情況。除此之外，虛擬資料必須包含許多情況，才能夠真正的幫助自駕車的訓練。

圖 5 為[10]中提到的訓練資料集，同一個場景中的 4 種天氣狀況。作者利用遊戲引擎易套入不同情境的優點，使同一個場景製作出清楚、灰濛、下雨、小雪等不同的效果。

圖 5.同一場景不同天氣狀況[17]

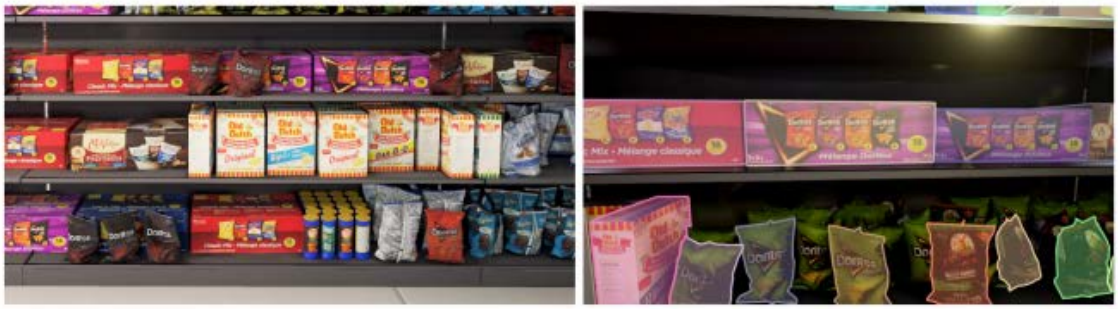


圖 6. UnrealCV 場景[11]

圖 6 左為利用 UnrealCV[11]輸出的超市場景，以作為訓練資料。而圖 6 右顯示，演算法成功辨識出真實世界的超市展架上不同的零食櫃。圖 6 左圖中的虛擬資料，成功騙過電腦，讓電腦認知為真實場景。

若遊戲引擎可以透過程式化的方式自動化調整鏡頭，就可以透過程式化的方式，取得場景資訊和場景變因(天氣、時間)，可以大量省去建立虛擬資料的時間。

[10]中，作者提到建立虛擬資料必須要有擴展性、使用方便、豐富的資源三個特性。而由於本論文的性質與[8][9][10]不同，我們依據這三個特性，找到最符合我們需要的建模軟體-SketchUp，而在 3.2 節本論文會針對 SketchUp 進行更詳細的描述。

2.2 貓狗二元辨識(混合訓練)

我們將真實世界的貓狗，與卡通、虛擬的貓狗圖片混合後，交給演算法訓練以及預測結果，圖 7 為實驗流程。在訓練階段，我們會在數據集中放入大量的虛擬貓/狗的照片，以及少量的真實貓/狗的照片訓練；在預測階段，我們把真實的貓/狗照片輸進訓練好的 CNN 模型，讓電腦辨識。

在訓練階段的數據集裡面，大量的貓/狗照片，可以代表本論文中產出的大量虛擬資料，而少量的真實貓/狗即代表本計畫中真實世界所採集的資料；在預測階段的數據集全為真實資料，是為了希望能夠將真實世界的房間照片給本團隊研究演算法預測後，能成功的畫出平面圖。

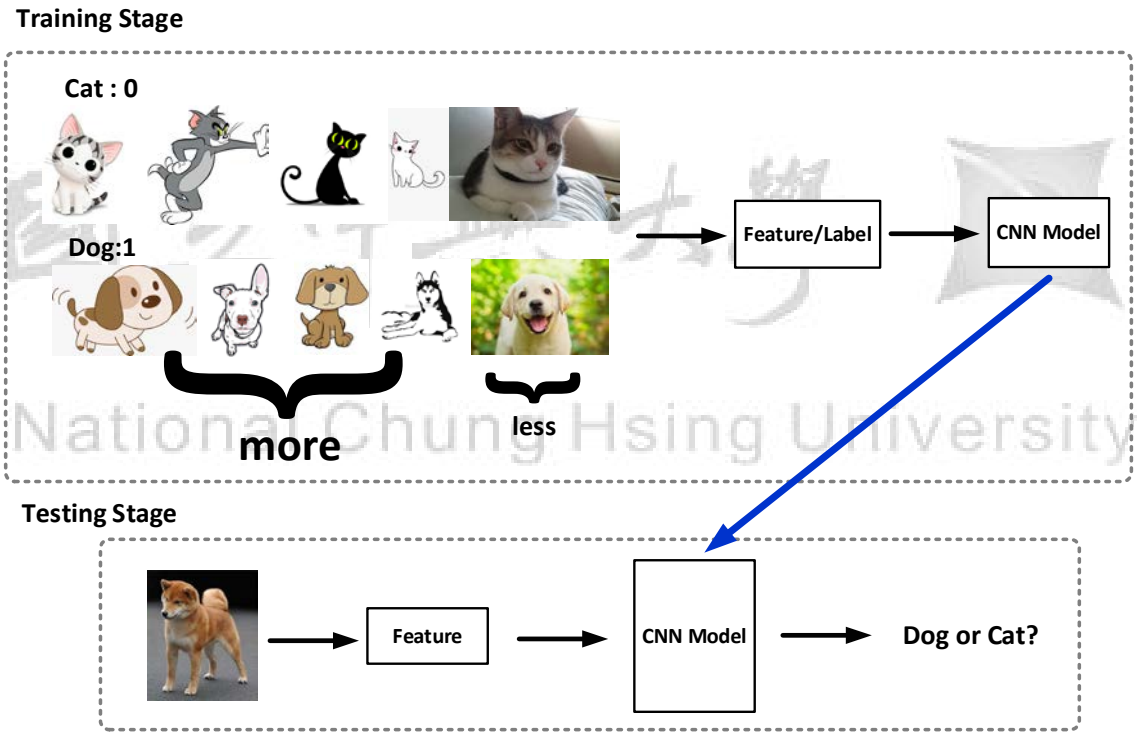


圖 7. 貓狗二元辨識機器學習架構

圖 7 中的數據集為透過 Google 搜尋圖片，進行網路爬蟲取得的照片，而 3.3.1.9 會介紹有關在 Google 網站取得照片的詳細介紹。我們篩選照片的依據為照片背景需乾淨，且只有一隻動物會出現在照片中，同時虛擬貓/狗的長相不能與真實的貓/狗長相差距太大。比方說擁有貓頭人身的卡通，以及 Hello Kitty、犬夜叉等，都不算在虛擬貓/狗的範疇。

圖 9 是圖 7 內的 CNN 架構，我們使用的 CNN 架構為 VGG16，在 VGG16

的卷積層與池化層，將原本 224*224 大小的照片產生數個子影像來縮減取樣。進入平坦層後，將原本的影像轉為 1 維長度 4096 個數值，最後加入 1 個隱藏層後，將 1 維長度 4096 個數值轉為另一個 1 維長度 4096 個數值。最後一層為輸出層，將輸出貓與狗 2 個結果。層與層間有 Dropout，即放棄部分神經元，以防止 overfitting。

在實務上我們常常在訓練模型時會遇到 overfitting 的問題，也就是模型在訓練階段時準確度很高，但是拿到預測階段時準確率卻異常的低。背後的主因是真正需要的模型比我們訓練出來的模型還要簡單。舉例來說，假設我們需要的模型是一條迴歸的直線，但是訓練出來的模型為了讓在訓練階段中錯誤率(loss)最小化，因此模型被訓練的結果不是我們想要的，這樣的模型拿去新的資料中錯誤率就會很高。通常有 4 種方法可以解決 overfitting 的問題，分別是改寫類神經架構、使用 L1/L2 正規化[4]、使用 Dropout，以及使用資料擴增，而在本論文主要使用 Dropout 及資料擴增的方式解決 overfitting 的問題。

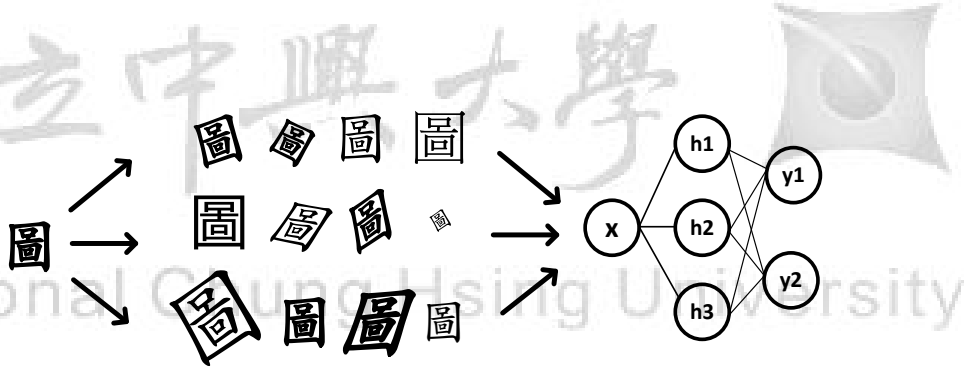


圖 8. 資料擴增

圖 8 展示了資料擴增的基本觀念。資料擴增是從既有的數據集中產生出更多的資料讓系統去學習，彌補資料不足的缺憾。雖然是假的資料，但也是從原始資料內容修改產生的，因此資料擴增經過證實，的確可解決資料不足的困境，並提昇系統訓練的準確率。

一張圖片經過旋轉、調整大小、比例尺寸，或者改變亮度色溫、翻轉等處理後，我們人眼仍能辨識出來是相同的相片，但是對電腦來說便是完全不同的圖像，因此我們可以將資料集中既有的圖片予以修改變形，創造出更多的圖片來讓演算法學習，彌補資料量不足的困擾。

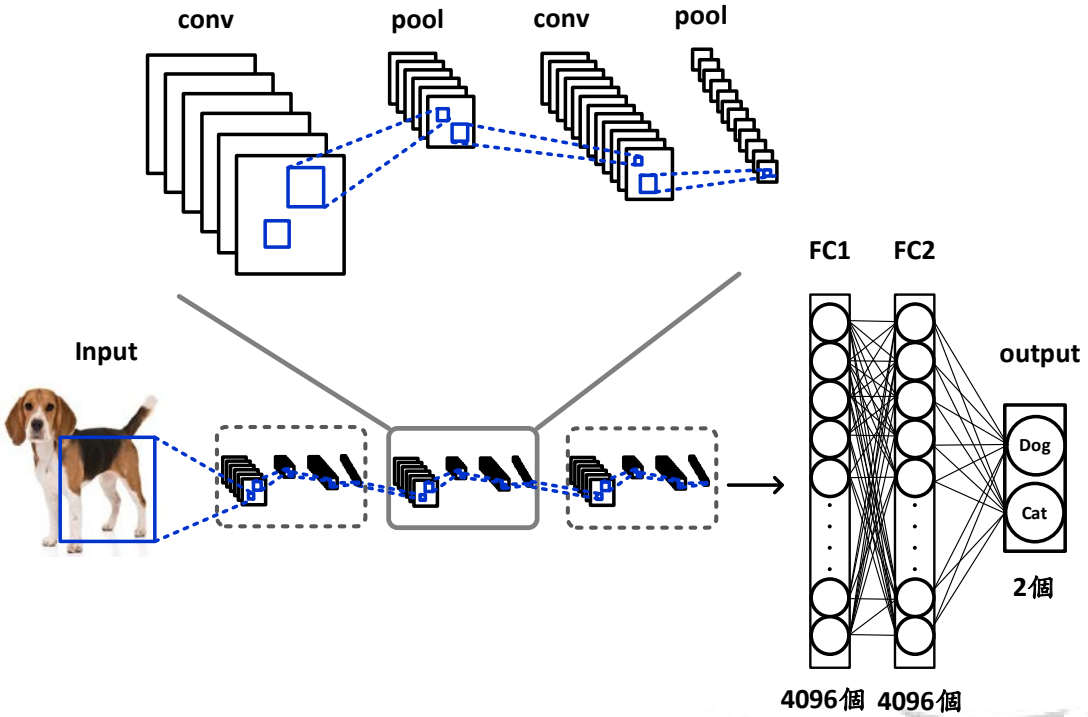


圖 9. 貓狗二元辨識的 CNN 架構

在整個神經網路的架構，我們需要思考及揣摩的地方很多，首先為隱藏層的層數，以及 Dropout 要放棄的神經元比例，若有過多的神經元會造成 overfitting，且收斂的速度會比較慢，反之若放棄過多的神經元，則會造成誤差變大等問題。除此之外，卷積層、池化層的層數，以及數據集的組成，也是貓狗辨識實驗中需要考慮的因素。

儘管貓狗辨識是監督式學習的二元分類問題，而本論文所探討的議題為監督式學習的回歸問題，且貓狗辨識的難易度也比本論文所探討的議題簡單很多，但藉由貓狗辨識，我們可以得知，我們的想法以及出發點是可嘗試的，本論文將實驗結果放在 4.1 節。

2.3 貓狗二元辨識(遷移學習)

遷移學習是深度學習領域中的一種訓練方式。若從現實層面考量，每一個人家裡或實驗室、公司的電腦並不一定都能夠處理龐大的訓練計算量，且現在市面上也有許多已經被訓練很好的演算法，故造就了遷移學習的廣泛應用。

為了達到高效率、高準確度的目的，除了經典的圖像辨識架構外，本論文也嘗試使用遷移學習，藉此觀察遷移學習如何幫助本團隊研究。

在唐朝的年代就已經有遷移學習的觀念，韓愈的《師說》裡面寫到「古之學者必有師。師者，所以傳道、受業、解惑也。」，同理，以遷移學習的概念，就是老師將所學的經驗傳授給學生，而以現在的時代而言，職場上的培訓、技術的複製也都有遷移學習的概念在裡面。遷移學習是將訓練好的演算法，用在目標演算法上，比方說我們欲訓練貓狗的二元分類，我們可以利用已經被 ImageNet[18]訓練過的演算法為基底，拿去訓練如何分辨貓狗。

我們拿 Task A 訓練的結果訓練 Task B，便是遷移學習的觀念。Task A 與 Task B 並無直接相關，雖然沒有直接相關，但也有一定的關聯性。在[5][13]中提到，TaskA 與 TaskB 可能為同域、不同種類，亦可能為不同域、同種類。以貓狗分類為例，假設欲做真實貓/狗的分類(Task B)，我們可以利用被真實大象/長頸鹿(Task A)訓練過的演算法進行遷移學習，或者利用虛擬貓/狗(Task A)訓練過的演算法進行遷移學習。

遷移學習的想法眾說紛紜，在[5]中提到，遷移學習領域中，我們將訓練 Task A 的數據集稱為 Source Data，而 Task B 的數據集稱為 Target Data，而不管是 Source Data 和 Target Data，都可分為 labelled 和 unlabelled 兩類，我們的焦點放在 Source Data 和 Target Data 都為 labelled 的情況下進行討論。在 Source Data 和 Target Data 都為 labelled 的情況下，一共有 Fine-tuning、Layer Transfer 和 Multitask Learning 三種方法，而我們主要是利用 Fine-tuning 和 Layer Transfer 的概念訓練貓狗的二元分類。

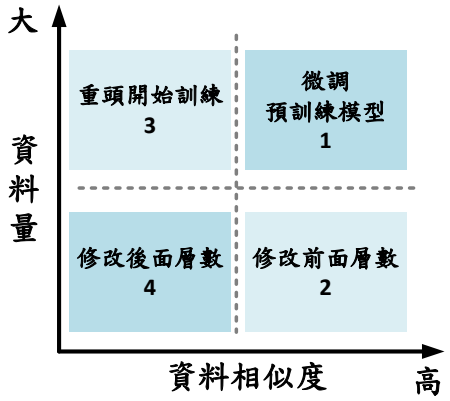


圖 10. 資料相似度/資料量對應使用方法

如[12]及圖 10 所示，我們依據資料相似度的高低，以及資料量的多寡分成 4 個部分討論。

圖 10 右上的 ①為數據集大，數據相似度高的情況。在此情況中，採用預訓練模型(pre-trained model)會變得非常有效。在 ①集大似高的情況中，最好的方式在保持深度學習模型原有的結構和初始權重不變，用新數據集再次訓練。

圖 10 右下的 ②為數據集小，數據相似度高的情況。在此情況中，因為數據與預訓練模型(pre-trained model)的訓練數據相似度很高，只需將輸出層改成符合問題情境下的結構，再重新訓練被影響的隱藏層即可。。

圖 10 左上的 ③為數據集大，數據相似度低的情況。在此情況中，因為我們有一個很大的數據集，所以神經網絡的訓練過程將會比較有效率。然而，因為實際數據與預訓練模型的訓練數據之間存在很大差異，採用預訓練模型將不適合，因此最好的方法是將預處理模型中的權重全都初始化後重新訓練。

圖 10 左下的 ④為數據集小，數據相似度低的情況。在這種情況下，我們可以固定預訓練模型中的前幾層的權重，然後重新訓練剩下的層數。因為數據的相似度不高，重新訓練的過程就變得非常關鍵。

在 4.1 節中，本論文分別利用虛擬貓/狗訓練的權重訓練真實貓/狗，及利用真實貓/狗訓練的權重訓練虛擬貓/狗，實驗結果一併在 4.1 節進行討論。

2.4 團隊研究目標

我們團隊的研究目標是，經手機建立座標資訊與相片，並利用這些資訊建立室內平面圖，如圖 11 所示。

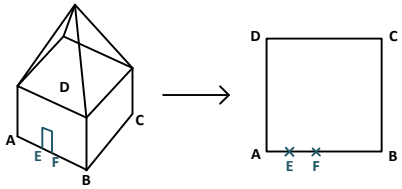


圖 11. 利用深度學習將室內空間進行分析並劃出平面圖

以圖 11 為例，深度學習演算法的輸入為標註過的室內空間影像，輸出為 A~F 點，我們再利用 A~F 畫出平面圖。我們的輸出為數個點，因此我們的研究目標屬於監督式學習的迴歸分析。本論文目標在建立深度學習所需的虛擬資料，在第三章會詳細介紹如何使用 SketchUp 進行資料的建立及虛實資料的併整。

因從現實世界取得資料不易，本論文會以 SketchUp 輸出的虛擬資料輔助本團隊的深度學習演算法。在 SketchUp 虛擬資料的建立流程，我們會先從 SketchUp 網站中抓取第三方的模型後，利用 SketchUp API 從模型取得相關資訊。

第三章 虛擬數據集

3.1 節會介紹團隊深度學習架構中所需資料集來源，而在 3.2 節會介紹本論文相關軟體，在 3.3 節為虛擬資料建立的方法。

3.1 數據集組成

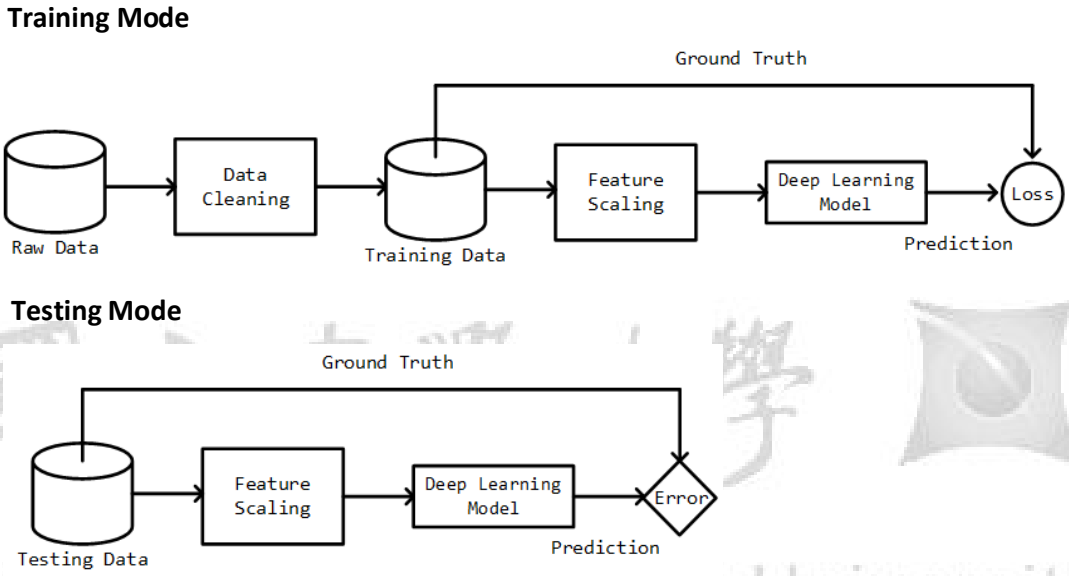


圖 12. 本研究團隊深度學習架構

圖 12 是本研究團隊的深度學習架構，我們將篩選過的資料提供演算法訓練，詳細的架構介紹請參考[2]。圖 13 為團隊研究的三種資料來源，分別為真實世界的資料、以真實世界為基底所建立之虛擬世界資料、虛擬世界資料。以下我們會分別針對圖 13 的三個部分進行討論，若需更詳細的真實世界資料描述，請參考[3]。

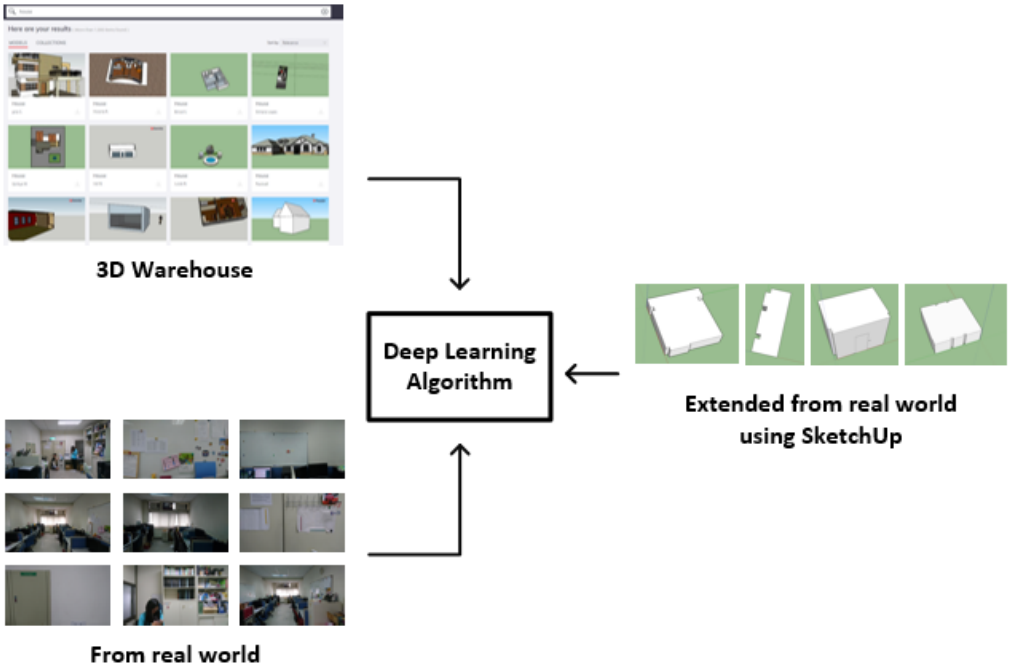


圖 13. 進入深度學習演算法的三種資料來源

3.1.1 真實世界的資料

真實世界資料取得方法為在一室內空間，手動量測房間資訊及門位置，接下來選 10 個拍攝點，並在每個點進行四個方向(各相差 90°)拍攝及紀錄。以後會運用手機慣性系統與錄影的方式，加快數據集的建立。

3.1.2　以真實世界為基底所建立之虛擬世界資料

本論文利用真實世界的資料，在 SketchUp 建置相似的模型，並且在每一個基於真實世界的模型中改變大小，成為新的數據。假如我們建立了 20 筆真實世界的資料，這 20 筆資料為母模型，我們將這些母模型進行延展後，儲存成新資料。我們將 1 個母模型，進行 30 次延展後，儲存成 30 個子模型，因此若有 20 筆的真實資料，我們便可以建立 600 筆新模型，提供演算法進行訓練。

SketchUp 中，若將模型令為元件(Component)，會出現如圖 14 的藍色長方體，長方體上有 8 個頂點，。透過 SketchUp 提供這些頂點，我們將元件進行延展，以建立成新資料。

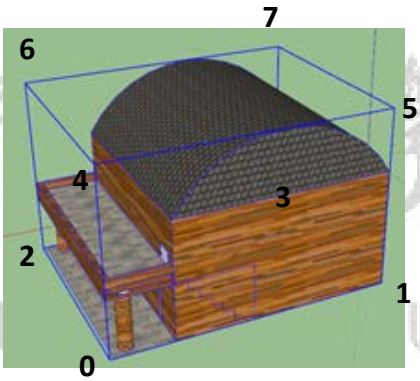


圖 14. 轉為元件

3.1.3 虛擬世界的資料

圖 15 為虛擬世界虛擬資料的建立流程，我們將圖 15 的流程圖分為 Before SketchUp、In SketchUp、After SketchUp 三大區塊。

Before SketchUp 是進入 SketchUp 的前置作業，首先我們會先從 SketchUp 的 3D Warehouse 中取得我們所需要的模型進行篩選後，將透過爬蟲取得的模型分為「建築物」與「家具」，若建築物的樣本數不高，也可以透過改變建築物大小，或者在建築物中放置不同的家具來將資料多樣化。

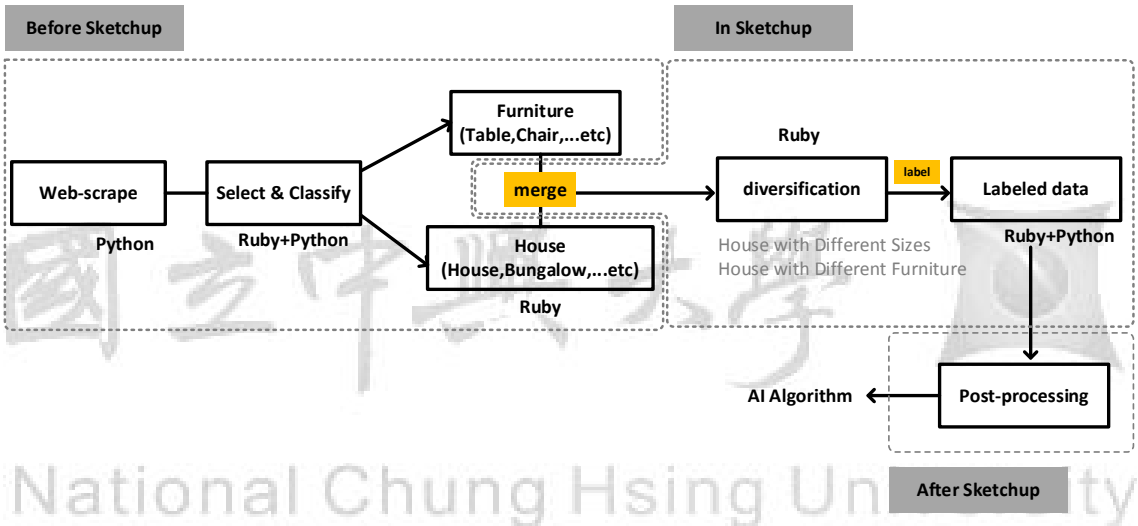


圖 15. 虛擬資料建立流程

在 In SketchUp 區塊中，我們透過 Ruby 程式語言，將下載的「建築物」與「家具」進行隨機的排列組合，等效即是資料擴增，以防止深度學習在訓練的過程中出現 overfitting 的問題。

在 After SketchUp 區塊中，我們利用自動化的方式將資料標註，並將資料轉換成演算法可以讀取的資料格式。同時將異常的資料進行刪除。

3.2 相關軟體介紹

3.2.1 SketchUp

我們使用 SketchUp2017-Make 作為建模軟體。目前 SketchUp 只提供 Windows 與 Apple 使用者使用，尚未開放給 linux 用戶。在 2017 版本中，Windows 提供了 SketchUp2017-Pro 與 SketchUp2017-Make，Pro 提供了完整的功能且終身可以使用，Make 則只提供繪畫功能及基本的功能。因考量用途及成本關係，我們使用 2017 年 Make 版本。

我們可以利用 SketchUp 提供的 Ruby API，進行點線面的生成、控制與取值，並將程序自動化。除此之外，由於 SketchUp 的使用者相當多，且為建模的主流軟體，故能下載較多的第三方模型。

市面上存在許多建模軟體，如: AutoCAD、Maya、Revit、SolidWork、Blender、Tinkercad 等。大部分的建模軟體只提供設計用途或動畫用途，鮮少提供研究用途，因此我們必須慎選軟體。表 1 為各軟體的比較表，最後說明選擇使用 SketchUp 的原因。

表 1. 建模軟體比較表

	程式語言	軟體	模型數量	上手程度	渲染	使用人數
SketchUp	Ruby	免費	多	中等	有-需購買	多
Blender	Python	免費	多	中等	有-免費	多
Solidwork	Visual C++	付費	多	難	有	多
Revit	Fortran	付費	多	難	有	多
AutoCAD	VisualLISP	付費	多	難	有	多
OpenSCAD	OpenSCAD API	免費	少	難	有	少

在表 1 中我們可以看出，SketchUp 和 Blender 有許多共通的優點，兩者都使用簡單、主流的程式語言 API，且使用者的人數也很多。但 SketchUp 和 Blender 之間也有許多差異。

我們選擇 SketchUp 作為建模軟體，是因為 SketchUp 的 3D warehouse 中的房間模型非常多樣。舉例來說，搜尋”House”關鍵字可找到的模型數(>14 萬)遠大於 Blender 能找到的數量 (< 100)，其他的建模軟體因為使用人數較

少，故下載的模型數會更少。SketchUp 的使用者多為建築相關領域，而 Blender 的使用者多為動畫、遊戲等，因用途不一，模型種類也不同，故 SketchUp 的模型較符合我們的需求。

而 SketchUp 的缺點是無法將渲染功能程式化，且渲染軟體成本較高，但 Blender 為遊戲引擎專用的建模軟體， Blender 建立的資料比起 SketchUp 建立的資料更逼真。

3.2.1.1 SketchUp 操作介面

圖 16 為 SketchUp 的操作介面，我們會運用 SketchUp 內的大工具集、Ruby Code Editor、功能表列、Ruby 命令列等功能。我們主要利用大工具集進行模型的微調，並透過 Ruby 命令列輸入程式觀看輸出結果。

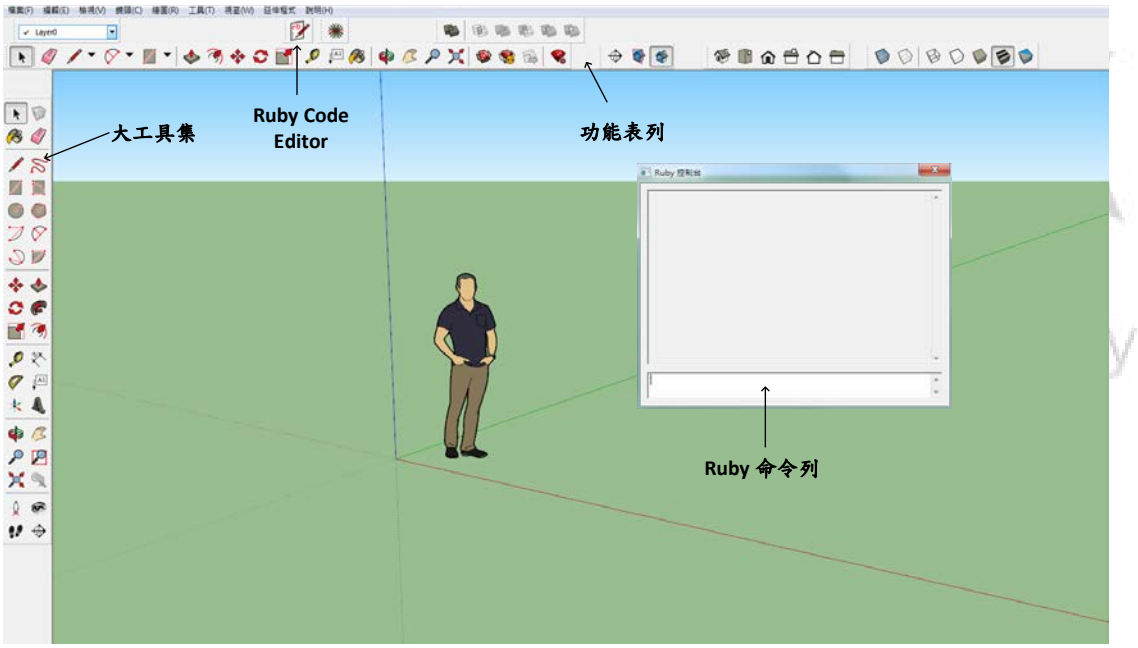


圖 16. SketchUp 的操作介面

圖 16 內的 Ruby Code Editor 是一種編輯工具，我們可以透過這個編輯器來撰寫 Ruby 程式碼，而 [1]有更詳盡的解釋，這裡便不多做說明。

3.2.1.2 SketchUp Ruby API

在 SketchUp 中，Ruby API 做為開發延伸功能的工具，許多人會使用 Ruby 撰寫外掛套件，以減少 SketchUp 建模時間。以下是 Ruby 帶給 SketchUp 的好處：

1. 在畫線及畫幾何圖形等工作，透過 Ruby 程式語言可以使繪畫更快速、更精準。
2. 當需要數學式運算，或者利用矩陣處理大量資料時，可以利用 Ruby 程式語言的內建函式庫完成人工繪畫做不出的功能。
3. 透過 Ruby 程式語言，可以將 json、yaml、csv、文字檔內的參數匯入 SketchUp 中使用，作為資料型態與 SketchUp 中間的溝通橋樑。

3.2.2 Python

在本論文中，我們使用 Python3.6 作為輔助。Python 為近期崛起的一門程式語言，除了有名的的機器學習、區塊鏈、大數據等用途以外，還可以用在資料科學、統計等各式各樣的領域。

Python 為直譯器(interpreter)，它不會一次把整個程式轉譯出來，而是每轉譯一行程式敘述就立刻執行，依序執行直到程式結束。直譯器的好處是它減少編譯整個程式的負擔，程式可以拆分成多個部分來模組化，但讓執行的效率變差，不過因電腦發展成熟，近年來能夠容許像 Python 等高階語言拖慢速度，卻可以因為 Python 簡單撰寫的特性，寫出功能更高、更齊全的程式碼。

3.3 虛擬資料建立

在 3.1.3 節中，我們將圖 15 的虛擬資料建立流程分成 Before SketchUp、In SketchUp、After SketchUp 三部分，而在 3.3 節中，本論文會針對 3.2.3 節做更詳細的介紹。接下來的 3.3.1 與 3.3.2 節，會先介紹 Before SketchUp 的部分。3.3.3 會介紹在 In SketchUp 中的操作流程，最後在 3.3.4 節中跟大家說明如何將資料轉換成深度學習可以讀取的資料型態。

3.3.1 爬蟲

在大數據的時代裡，資料集非常重要，資料科學家在拿到相關資料集後，便可以做一些演算及分析。深度學習研究主題非常多樣，舉例來說，分子生物學方面，資料集源自於一群花很多時間在研究的研究生以及科學家；醫學方面，資料集源自於醫院病人的病歷。

網路也是現今必備的工具，現今在網路上有許多的資料，而從網路上索取大量資料的動作，稱為「爬蟲」。因應大數據時代的來臨，爬蟲技術普及化，原本在爬蟲的世界中，我們應該要懂的網路協定，現在有許多人寫出的套件包都幫我們做好了，如 Python、Ruby、Javascript…等等，支援了許多套件包，方便我們能夠完成爬蟲技術，我們不再需要花很多的力量理解並實現爬蟲，只需要短短的程式碼即可完成，而以下將會介紹基本及常用的爬蟲函式庫 Requests、Beautifulsoup4、Selenium。我們可以使用 Requests 在網頁爬蟲，可以自動提取網站上的數據信息，並把這些訊息利用 Beautifulsoup4 以一種容易理解的格式呈現出來，或者使用 Selenium 控制瀏覽器幫我們爬取資料。

3.3.1.1 網站的組成

網站是由 Html(HyperText Markup Language)、CSS(Cascading Style Sheets)、Javascript 三部份所構成，Html 主要代表架構，而 CSS 和 Javascript 分別代表網站的外觀與行為。

一般網站區分為靜態網站與動態網站二種，靜態網站指的是，無法透過管理介面讓管理者即時新增、修改或刪除的網頁內容。反之動態網站則是可以讓管理者透過後端管理系統即時的針對網站進行更新或維護的動作。動態網站的結構，可分網站前端與後端。動態型網站的資料，必須先由網站後端

(網站後端管理系統)將資料與相關文件上傳或做設定後，便會即時的顯示於網站的前台供網路使用者瀏覽 [16]。

3.3.1.2 Ajax 網站

Ajax 即 Asynchronous Javascript And XML[15]，是指一種建立互動式網頁應用的網頁開發技術。Ajax 意即非同步 JavaScript 和 XML。Ajax 也是一種用於建立快速動態網頁的技術，且是一種在無需重新載入整個網頁的情況下，能夠更新部分網頁的技術。雖然其名稱包含 XML，但實際上資料格式現今都已由 JSON 代替，進一步減少資料量，形成所謂的 Ajaj。本論文爬取的 3D warehouse 網站，是使用 JSON 格式。

JavaScript 可以向伺服器發送請求，並解析、響應。除此之外，JavaScript 也具有更新網頁的能力。熟悉 JavaScript 的開發者可以做到只更新部分的頁面，而毋需向伺服器索取整個頁面。當在 Ajax 進行動態載入，使用者在頁面上觸發事件，javascript 會對伺服器發出一個請求，並把資料渲染回客戶端。他的優點是觸發資料載入的當下，使用者還是可以繼續使用該程序。舉例來說，現今許多知名網站如 Facebook 的網頁內，會隨著卷軸不斷往下拉時，出現新東西，而此類即為 Ajax 網站。

在動態載入的網頁中，爬蟲過程不能在網站上觸發特定事件，我們便拿不到需要的資料。這個時候會有兩種思路，第一種是想辦法模擬使用者在瀏覽器上的行為，進而觸發資料的載入。常見的方法為 Selenium、PhantomJS、CasperJS 等等，而另一種是觀察登入時的 POST request，觀察 javascript 在客戶端向伺服器請求了哪個資料檔，我們最後爬取 3D warehouse 網站的方式屬於第二種，在 3.3.1.8 會介紹。

3.3.1.3 Requests

Requests 模組讓我們容易從網站上下載檔案，不必擔心網路錯誤、連結出錯以及資料壓縮等複雜問題。Requests 支援 HTTP 保持連線狀態、支援使用 cookie、支援檔案上傳、支援自動確定響應內容的編碼…等，而 Requests 不是 Python 內建模組，所以要另外安裝。

如果想要使用 Python 來下載網頁上的資料，最基本的作法就是以 Requests 模組建立適當的 HTTP 請求，透過 HTTP 請求從網頁伺服器下載指定的資料。

以下先解釋 Requests 的 GET 與 POST ，此兩名詞會在 3.3.1.8 爬蟲時用到。

當使用者在網站使用 GET 請求伺服端，伺服器端回傳資料。傳回資料的網址(URL) 會隨著不同的網頁改變。假設 HTTP 為信封撰寫格式，信封外的內容稱為 http-header，信封內的書信內容為 message-body， HTTP Method 就像寄信規則。GET 及 POST 就是 HTTP Method 的兩種方法。

GET 表示信封內不得裝信件的寄送方式，如同是明信片一樣，我們可以把要傳遞的資訊寫在信封(http-header)上，使用 GET 的時候我們直接將要傳送的資料加在我們要寄送的網址(URL)後面即可。

圖 17 為 Google Chrome 內建的 Google 開發者工具，透過開啟 Network 選擇 xhr 點選 Headers 後出現的畫面，我們可以在開發者工具面板上看到的請求，多為 GET 和 POST。

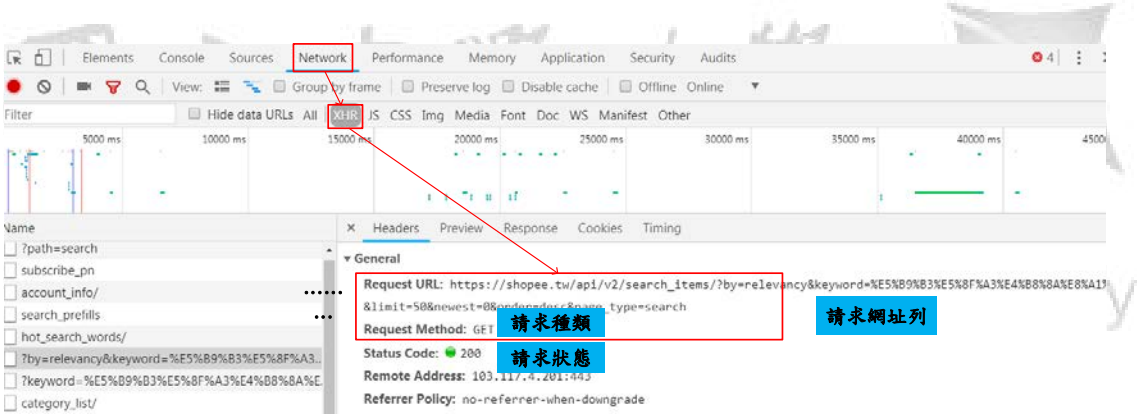
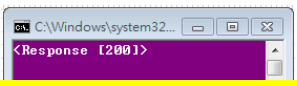


圖 17. Google 開發者工具

以圖 18 為例，當我們使用 GET 請求，再加上適當的 Headers，最後讀取到狀態碼為 200，即為請求成功。

GET

```
url = "https://shopee.tw/api/v2/search_items/?by=relevancy
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; Win64; x64)
}
res = requests.get(url,headers=headers)
print(res)
```



200 OK

請求成功。成功的意義依照 HTTP 方法而定：
GET: 資源成功獲取並於 message body 內發送。
HEAD: entity 標頭已於 message body 內。
POST: 已傳送 message body 內的 resource describing the result of the action。
TRACE: 伺服器已接收到 message body 內含的請求訊息。

圖 18. 使用 Requests 函式庫-GET

因現今爬蟲技術盛行，有些大型的公司如 Google、蝦皮拍賣，提供 API 給爬蟲使用者使用爬蟲擷取資料，而有些網站則會使用「反爬蟲」機制防止爬蟲程式侵入。反爬蟲機制會分辨是否為爬蟲程式對網站發出的請求(Request)，因此許多爬蟲使用者會透過加 Headers、cookie 等各方法來設法偽裝成非爬蟲使用者。若寫入的 Headers 無法請求成功，我們可以使用 Selenium 破解，相關介紹會在後面詳述。

POST 就像信封內裝信件的寄送方式，不但信封可以寫東西，信封內(message-body) 還可以置入欲寄送的資料或檔案。訂高鐵票的網站為使用 POST 最典型的例子，圖 19 是 POST 運作的示意圖，我們將出發站、到達站、搭車日期...等資訊填妥後，利用 POST 請求傳送出去。

圖 19. 以訂高鐵票的網站解說 POST 運作模式

圖 20 是以爬取高鐵站網站為例的腳本(script)，我們將出發地點、抵達地點...等資訊從 Google 開發者工具複製到腳本內的 Form_data 中，執行腳本後，再用 Pandas 工具將回傳的資訊表格化。Form_data 的內容，好比信紙內容，將信紙放進信封內丟出去。Pandas 的輸出結果為圖 20 紫色區塊的 10 筆車次資料。

圖 20. 爬取訂高鐵票網站的程式碼及輸出結果

在 POST 中，除了爬取上述高鐵票網站時，是以 Form Data 的方式傳給伺服器，另一種格式是 Request Payload。

在對 Ajax 做出請求時，若發送的資料格式為 Form Data，此時 Headers 的 contentType 為 application/x-www-form-urlencoded；若發送的資料格式是字符串(json)，則為 Request payload，此時 Headers 的 contentType 為 application/json。

3.3.1.4 BeautifulSoup4

BeautifulSoup4 是一個 Python 的函式庫模組，可以讓開發者僅須撰寫非常少量的程式碼，就可以快速解析網頁 HTML 碼，從中取出使用者有興趣的資料，降低網路爬蟲程式的開發門檻、加快程式撰寫速度。

BeautifulSoup4 的運作方式是讀取 HTML 原始碼，自動進行解析並產生一個 BeautifulSoup4 物件，此物件中包含了整個 HTML 文件的結構樹，有了這個結構樹之後，就可以輕鬆找出任何有需要的資料了。

在 BeautifulSoup4 解析網頁上，大致上可分為以 HTML 屬性搜尋、以 CSS 搜尋、以文字內容搜尋、以某位置為基準向上向前與向後搜尋...等功能使用，以圖 20 為例，我們將 BeautifulSoup4 結合 Pandas 表格工具，利用圖表的方式統整出有系統的資料。除此之外，BeautifulSoup4 也常常搭配正規表示法(Regular Expression)使用，利用 BeautifulSoup4 抓取我們要的 Html 標籤，再利用正規表示法取得想要的關鍵字或數字。

在 2018 年 3 月，[14]提到 Requests 內部提供可以解析 html 的新項目 Requests-HTML，因此解析 HTML 多了一個新選擇。現在許多網站不會將全部的資訊放入 Html 中，可能會以 javascript 的方式寫入資訊，其存取資訊的資料格式為 json，而 BeautifulSoup4 卻無法處理 json 檔。即便如此，BeautifulSoup4 在現今仍還是解析 html 最方便的軟體。

3.3.1.5 Selenium

Selenium 控制瀏覽器來進行網站操作的自動化，它能夠直接獲取即時的內容，包括被 JavaScript 修改過的 DOM [6] 內容，讓程式可以直接與網頁元素即時互動、執行 JavaScript 程式，因此也適用於前端採用 Ajax 技術的網站。

Selenium 是許多網路測試工具的核心，利用 Selenium 操作網頁表單資料、點選按鈕或連結、取得網頁內容並進行檢驗，可以進行多種類的測試。Selenium 發展迄今，已有四個主要的專案發行，分別為 Selenium IDE、Selenium Remote Control、Selenium WebDriver、Selenium Grid。目前我們只用到 Selenium IDE 及 Selenium WebDriver 進行爬蟲。

因 Selenium WebDriver 可以模擬人對於瀏覽器所有的操作行為，所以最不會被防制爬蟲軟體所偵測到。

在 Selenium IDE 中可以使用錄製功能，當使用者於瀏覽器上面任何操作，都會被 IDE 紀錄，並可以轉為 Python 或其他腳本語言，未來若要執行一樣或類似的動作，只需將原來的腳本進行修改即可。

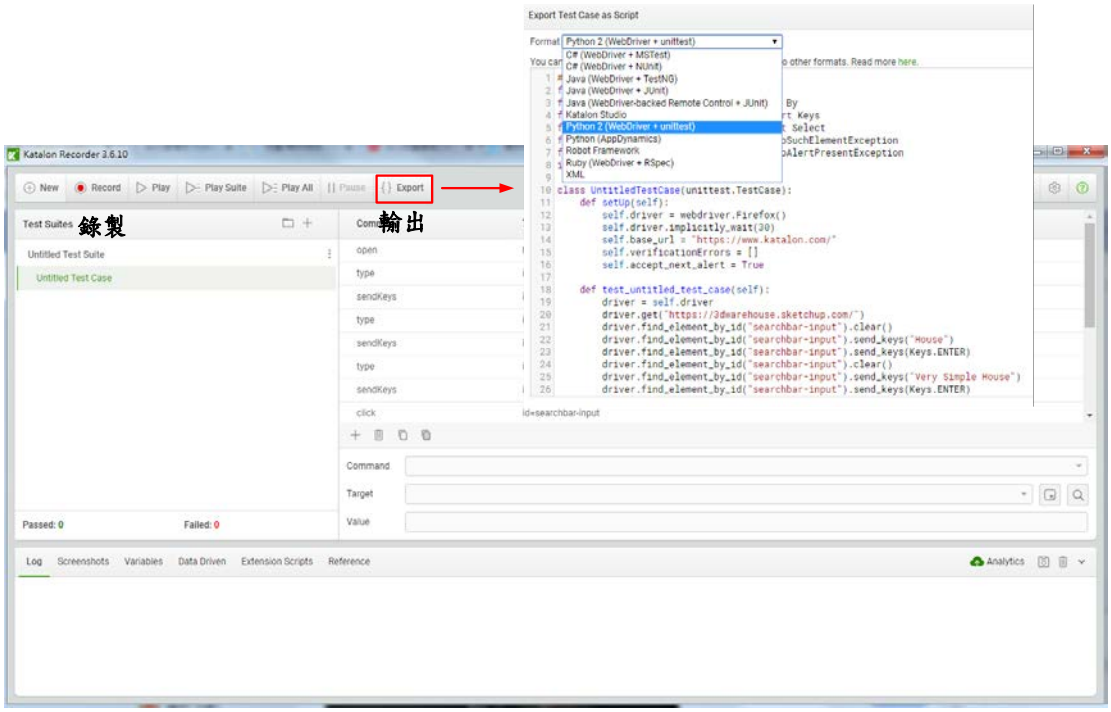


圖 21. Selenium IDE 介面

圖 21 是 Selenium IDE 的介面，對於初學者來說是最方便的 Selenium 編寫工具，我們只要按「錄製」，Selenium IDE 就會將我們使用瀏覽器的動作記錄下來，像是按什麼按鈕、打什麼字、登入什麼頁面...等等，記錄下來以後可以將這些動作輸出成 Python、Ruby、C#...等等各程式語言的腳本。

在伺服器上使用 Selenium 來做網站資料擷取，與使用者親自電腦執行不同的地方，在於使用者能夠讓 Selenium 控制的瀏覽器新增視窗運行，而伺服器則不能。這可能會拖慢程式執行，為了解決這樣的狀況，Selenium 可以指定以 Headless 模式(無頭瀏覽器)運行，不用開啟瀏覽器視窗也可以運作。如此可以給無法開啟瀏覽器的作業系統、伺服器一項解決方案，且執行的速度也比開啟瀏覽器方式還要快。

寫入適當的 Headers 非常的重要，有些網站為了防止爬蟲進入，所以每次登入進去的 Headers 都需更換。若遇到上述狀況，也可以使用 Selenium 嘗試去突破難關。

```python
import requests
import json
import pandas as pd
from bs4 import BeautifulSoup
from selenium import webdriver

# 無頭瀏覽器進入
options = webdriver.ChromeOptions()
options.add_argument('headless')
driver = webdriver.Chrome(chrome_options=options)
driver.get('https://www.programiz.com/python-programming/generator')

# 取headers
cookies = ['{}={}'.format(item.get('name'),item.get('value')) for item in driver.get_cookies()]
cookies = ';'.join(cookies)
token = [item.get('value') for item in driver.get_cookies() if item.get('name')=='csrftoken']
print(cookies)
print(token)
driver.quit()
headers = {
    'Referer':"https://waste.epa.gov.tw/WasteConfigure/VocationCode.asp",
    'Cookie':cookies,
    'Content-Type':'application/json',
    'x-csrftoken':token[0],
    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
}


jd = json.loads(r'[null,null,"\"3HTwf81aKSXQ5le9tiVXWl9g2iPh5nDhhhZ23RIn9UMKwlcDxQDK74m6OKTqEIYAb,
res = requests.post("https://shopee.tw/__t__",json=jd) #headers=headers)
print(res)
```

圖 22. 爬取蝦皮網站腳本範例

在 3.3.1.3 提到，有些爬蟲在爬取網站時，需要有 Headers 偽裝瀏覽器登入才能登入成功，而 Headers 的相關參數只能自己去測試。有些網站在請求時不需要 Headers 即可登入，有些可能需要少量的 Headers 參數，甚至有些需要非常多的 Headers 參數才可以登入成功。

圖 22 是一個爬取蝦皮拍賣網站中的一段腳本，當我們要在蝦皮拍賣中爬取物品的資訊、價格...等等訊息時，網站提供 POST 的方式讓我們爬取。對蝦皮網站的請求的手續比較繁瑣，我們需要 Referer、Cookie、Content-Type、User-Agent、x-csrftoken 等參數才能登入網站。因蝦皮網站在反爬蟲上的機制較複雜，在抓取 Headers 相關資訊時，每一次抓取的訊息都不一樣，無法直接進入 Google 開發者工具來取得 Headers 相關訊息進行登入，於是我們用 Selenium 的方式進行登入。當開啟瀏覽器，就會載入 Referer、Cookie、Content-Type、User-Agent、x-csrftoken 等等資訊，我們藉由瀏覽器載入的動作時產生的參數，抓取 Cookie 跟 Token 兩參數，利用此兩種參數進行網站的登入。

3.3.1.6 多執行緒

在[7]中提到了大量多執行緒的語法，因此本論文不再贅述，我們發現執行單執行緒(1 個主執行緒)時，會花費大量時間，若利用多執行緒執行耗費的時間會縮短，如此可以減少資料蒐集時間。以下的程式功能是將命令列輸

入多個關鍵字，這些關鍵字在 3D warehouse 的網站分別找尋相關的模型檔(.skp 檔)。舉例來說，當本論文 3.3.1.8 節中的第一版爬蟲程式輸入 house,sofa,chair，程式會先搜尋 house 相關的模型，再來搜尋 sofa 相關模型，最後搜尋 chair 相關的模型檔案。若以單執行執行，需等到搜尋完 house 以後，才會開始執行搜尋 sofa，最後搜尋 chair 相關的模型，以這種執行方式會耗費很長的時間。因此，本論文其後使用多執行緒，可以同時執行尋找 house、sofa、chair 的動作。圖 23 是以 Requests 和 Beautifulsoup4，以多執行緒的方式執行的例子。圖 23 中的下方紅色框為多執行緒的方式執行 job() 的工作， t=threading.Thread(tarGet=job,arg=(url,))及 threads.append(t)串起所有的 job()工作。

```
1  # 網址：https://hk.saowen.com/a/d73234579ed55bda4f8775f0058b09b5e7b5a4
2
3  def job(url):
4      from selenium import webdriver
5      driver = webdriver.Chrome()
6      driver.get(url)
7      driver.quit()
8
9  def multithreading():
10     import os
11     import threading
12     import time
13     str_Key = input("Input Keyword([key],[key],[key],...):")
14     str_Key = str(str_Key)
15     folder_array = str_Key.split(',')
16     str_Key = str_Key.replace(' ', '%20')
17     str_array = str_Key.split(',')
18     str_index = len(str_array)
19     str_front = "https://3dwarehouse.sketchup.com/search/?q="
20     str_back  = "&searchTab=model"
21     threads = []
22     for item in range(str_index):
23         str_array[item] = str_front + str_array[item] + str_back
24     url_tuple = tuple(str_array)
25     for url in url_tuple:
26         print(url)
27         t = threading.Thread(target=job,args=(url,))
28         t.start()
29         threads.append(t)
30     for thread in threads:
31         thread.join()
32
33 if __name__=='__main__':
34     multithreading()
```

工作內容

同時執行
工作內容

圖 23. 多執行緒的程式範例

3.3.1.7 解析 3D Warehouse 網站

平時爬取網頁時，可能會遇到一種情況，當我們以 Requests 爬取網頁時，網頁請求得到的 HTML 中並沒有我們在瀏覽器中看到的內容，會出現這種情況，是因為這些資訊是透過 Ajax 載入，經由 javascript 渲染生成。而 SketchUp 的 3D warehouse 網站運作的性質即是如此。

由於 Ajax 的網頁技術興起，許多網頁框架的寫法往往改為將 Html 的內容精簡化，利用 javascript 載入真正的內容。爬取網頁前，可以在 Google 開發者平台進入 Network→Doc，或者按滑鼠右鍵以檢視網頁原始碼的 Html 內容。若進入 Network→Doc 找不到內容，我們可以從 Network→XHR 尋找.js 檔，並查看.js 裡的內容。Ajax 一般是通過 XMLHttpRequest (簡稱 XHR) 物件介面傳送請求，簡單來說，Ajax 網址在網頁載入時，瀏覽器位址列的網址沒有變，而卷軸不斷的往下拉的同時， javascript 一邊載入網頁。我們在 Chrome 中開啟開發者工具→點選 Network →點選 XHR 標籤，此時我們就可以看到 XHR 標籤，如圖 24 所示。

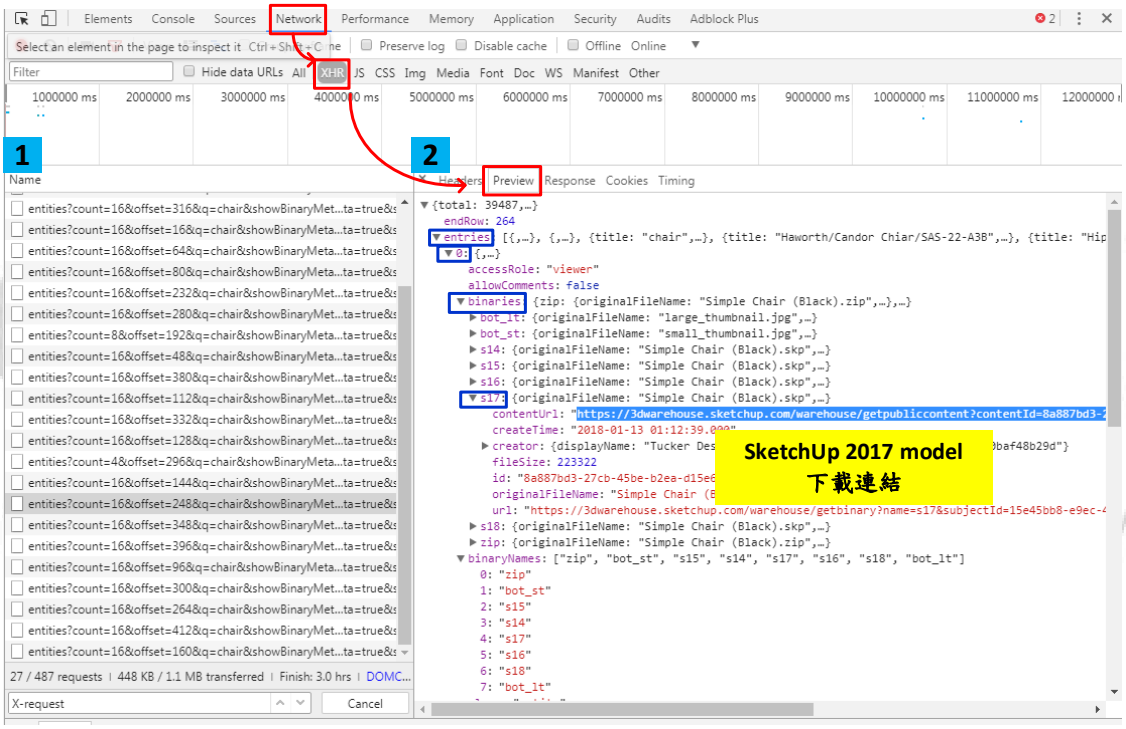


圖 24. 利用 Google 開發者工具尋找模型的下載連結

在 3D warehouse 中，當頁面一直往下拉時，圖 24 左的 javascript 區 1 ，會不斷地新增新的 xhr 請求，而右邊便是 xhr 標籤的詳細內容，而在 Preview 區→ entries → 0 → binaries → s17 → contenturl 對應的網址就是我們想要的模型下載連結。

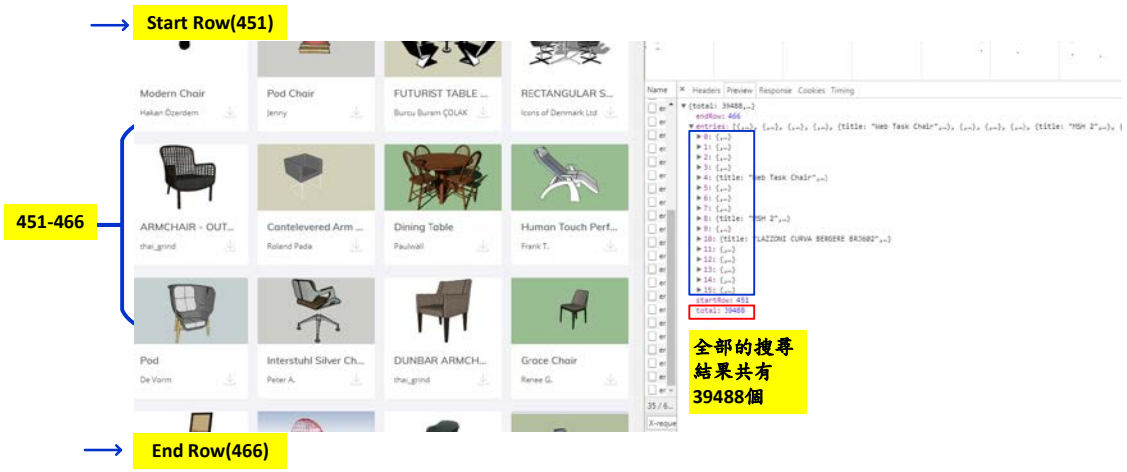


圖 25. 在 3D Warehouse 開啟 Google 開發者平台的說明

圖 25 說明在 3Dwarehouse 網站中，xhr 內容與搜尋網頁結果的對應關係。圖 25 右的 xhr 顯示了「椅子」的搜尋結果共有 39488 個，而每次 xhr 標籤共請求顯示 16 個(右邊藍色框)，此 xhr 標籤請求的是第 451~第 466 個。而圖 25 左是這 16 個對應的椅子。

本論文介紹爬取 3D warehouse 的兩種寫法，第一種是運用 Selenium 及多執行緒的方式爬取，它不需要太多的爬蟲基礎即可使用；另一種為觀察當 Ajax 載入網站時給予伺服器端何種請求，並藉由這個請求找尋出可以下載第三方模型的方式。第二種方法，我們最後找到一個可以找到模型連結的 GET 請求，並找出規律後，將網路上大量的模型下載。

3.3.1.8 爬取 3D Warehouse 網站

圖 26 為使用 Selenium 爬蟲的方塊圖，首先先輸入多個關鍵字搜尋，假設輸入的關鍵字 house,bungalow,chair，這些關鍵字會分別從 3D warehouse 中搜尋 house、bungalow、chair 的頁面中尋找模型，透過連結取得下載網址，下載至客戶端。

若我們輸入 5 個關鍵字，會分別進入到 5 個不同關鍵字的頁面，請求 5 次。若每一個頁面都有 1 萬個模型，總共加起來要取出 5 萬個連結，且每個連結中又要取得另一個下載連結。跑完上述流程就需約 10 萬次的請求，對網路來說負擔很大。實際上在爬取時不僅網路流量變得很小，程式執行不完的問題之外，真正能夠載下來的數量也不多，因此我們需要改善。「第二版爬蟲腳本」是基於 Ajax 特性所寫出來的腳本，第二版的腳本請求次數較少，且效率高很多。

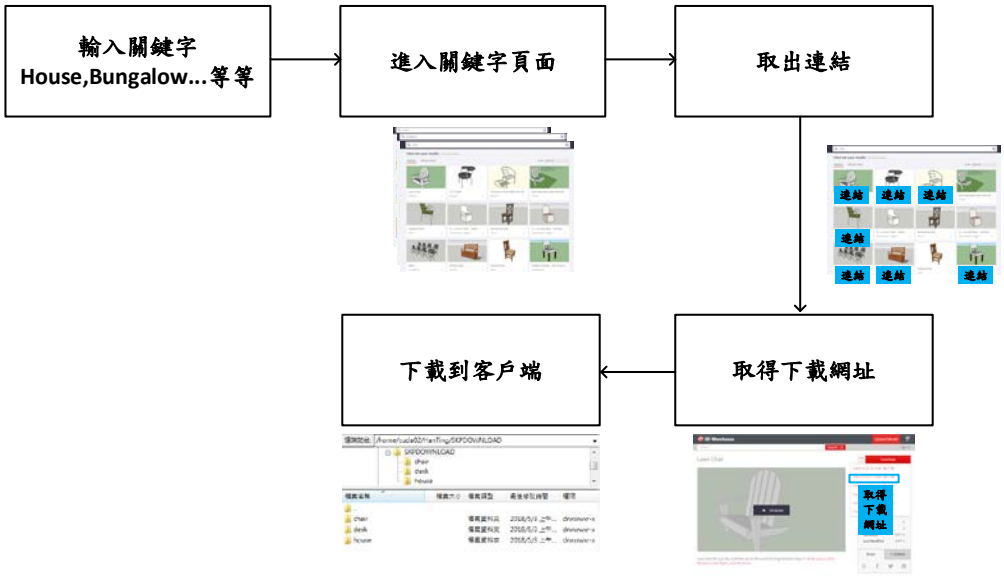


圖 26. 第一版爬蟲腳本流程

圖 27 是利用 Ajax 特性寫出的腳本方塊圖。利用 Ajax 特性寫出來的結果比起 Selenium 更為方便且有效率，並且腳本內容更精簡，少了「取出連結」至「取得下載網址」的手續，讓程式的執行更直觀，向伺服端請求的次數更少，且更關鍵的是，利用 Requests 取得的請求，比起 Selenium 開啟瀏覽器的方式，速度更快，也不易有當機的問題。

圖 27. 第二版爬蟲腳本流程

圖 28 說明 json 的網址說明與 json 的頁面。Json 檔格式與 Python 的 dictionary 相似，因此比起 html 還要好理解。

圖 28. 3D Warehouse 網站中取得的 GET 網址列說明

3D warehouse 網站的請求方式為 Get，以圖 28 放大處的網址列為例，count 代表可以載入 json 檔的網址數目，而 offset 有定位功能，代表從第幾筆資料開始，若 offset=16&count=16，代表從第 16 筆資料開始下載 16 筆資料，意即從第 16 筆下載到第 31 筆。而經嘗試發現 count 最大容許值為 1000，所以若要下載 10 萬筆資料左右，只要向 3D warehouse 網站請求 100 次的 json 檔，即可以取得下載連結，比起使用的 selenium 請求的 10 萬次相比效率高非常的多。

3.3.1.9 爬蟲 API

3.3.1.3 提到，有些網站會提供官方 API 提供給使用者爬取，如 Facebook、Instagram、Google 等知名網站。除了官方提供的 API 以外，常常也有許多第三方的 API 提供使用者使用。以下介紹[17]如何用 Google 的 API 批量下載 Google 圖片。

[17] 是一個第三方的 API，首先在命令列打”pip install google_images_download”，即可使用。在使用上面，舉例來說在命令列打”googleimagesdownload -k cat -l 10”，其中 -k 指的是 "keyword"，也就是關鍵字，而-l，為"limit"，指的是圖片的數量限定，所以上述的例子指的就是下載 10 張貓咪的照片。

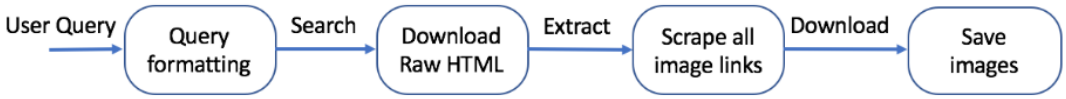


圖 29. Google 圖片爬取的第三方 API 爬取流程

如圖 29 所示，在[17]API 的程式演算法流程和 3.3.1.8 所提出來的第二版爬蟲腳本流程極為相像，只不過[17]中由 Html 解析出下載連結，而第二版的爬蟲腳本使用 json 取得下載連結。

而在[17]提供的 API 中，因防止占用伺服端網路容量，作者限定了最多只能下載 100 張，然而對於要下載大量照片來說，是非常不利的。然而我們不用修改底層，API 中提供了另一項服務，運用 Selenium 開啟多視窗下載批量圖片，且由於是用無頭瀏覽器開啟，所以不會造成電腦畫面的混亂，也不至於過於影響電腦速度。

我們透過[17]，可以運用最快的速度下載我們需要的照片，應用在 2.2 貓狗的分類。現在還有很多有用的 API，如 Instagram 的 API 可以下載來自不同地方、不同社群的照片等[19]。

3.3.1.10 資料如何影響深度學習

我們都知道，深度學習的主角並不是程式本身，而是資料。不像傳統的程式撰寫，我們不用花費大量的人力想著要如何寫不同的條件因應不同的資料，取而代之的是利用資料訓練演算法，讓演算法能夠透過這些資料預測。

在訓練演算法的過程中，我們容易在蒐集資料的同時，帶有資料本身的偏見。當我們要訓練貓狗辨識的模型，可能因為所下載的貓和狗的種類並不那麼地齊全，這些貓或狗並沒有辦法代表全世界的貓與狗，因此這些我們所下載的資料，本身就有類似人類偏見的存在。所以在本論文題目中，我們預下載的 SketchUp 模型，即便再多，都有可能隱含挑選所產生的偏見，而真實的資料更是如此。因為地域性的關係，以及資料取得不易，我們大多挑選的資料來源為學校的教室，因此當要預測學校以外的建築物，挑戰性會大。

隱含的偏見也會容易影響最後所預測的結果，舉例來說，如果我們要訓練一部電腦去學習辨認一隻貓，若我們在訓練演算法時都用摺耳貓來訓練，我們的運算法則最後會出現偏向貓耳是捲起來的隱含偏見。

總而言之，最後深度學習所預測下來的準確度，跟許多環節環環相扣。可能 A 資料出來的是較好的結果，而 B 資料產生的是較不好的結果，除了自身的深度學習演算法架構外，資料本身所含的偏見也會影響整體演算法預測的表現，這也是非常值得探討的議題。

3.3.2 爬蟲後的模型篩選

爬蟲後的 SketchUp 模型篩選，是由人工判斷資料可否使用。雖然很耗時，卻很重要。本論文一共爬了 14 萬筆的資料，經篩選後，只剩下幾十棟左右。

在爬蟲的階段，SketchUp 的 3D warehouse 網站上放著來自世界各國的作品，任何人都可以將自己作品上傳，3D warehouse 的網站不會篩選作品，所以在可能會出現太簡陋的作品，甚至是半成品，因此不能使用的模型非常多。我們篩選資料的依據為「簡單」、「建築物內沒有房間」、「門位置容易找尋」的原則，以關鍵字若為”house”、”building”…等等越廣義的單字，越能搜尋到更多筆資料，但是被淘汰的資料也越多，導致花大量時間在篩選，降低篩選的效率；相反地，如果關鍵字為” bungalow”、”classroom”、”lab”…等等較狹義的關鍵字，便可以減少篩選時間。

本論文旨在「在模型內拍照」、「建立房間資訊」、「建立門資訊」、「建立照片資訊」。本論文必須透過分析房子，得出一套經驗法則來設計對應腳本，而經驗法則以外的資料，只能人工標註模型的房間資訊、門資訊。

3.3.3 In SketchUp

爬完、篩選完資料後，正式進入 SketchUp 的程式部分。首先將抓取下來的模型實體(entity, or ent)中的面抓取下來分為垂直面與水平面，分別取出門與房間資訊，抓取到目標面進行分析後，取出我們需要的數值，以及在模型內拍攝的照片。以下我們會用面處理、房間資訊、拍照資訊、門資訊 4 個小節更細部介紹主體部分的運作，最後進行總結。

3.3.3.1 面處理

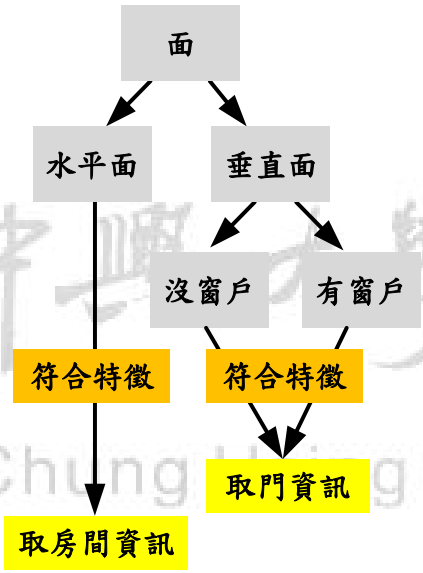


圖 30. 面處理流程

如圖 30 所示，本論文將模型中的面取垂直面與水平面，取水平面是為了日後取得「房間資訊」，而取垂直面是為了日後取得「門資訊」。

在 SketchUp 中取得「面」需要一點小技巧。在 entity 的 Drawing Element 類別中，元件實例(Component Instance)與面(Face)其實是不同的類別，所以就算在元件實例中內包含許多面，我們也無法任意取用元件實例內所包含的面。

在完成的作品裡可能有非常多的部件，有門、窗、電視、桌椅...等許多家具。SketchUp 的使用者們常為作畫方便，使用功能將畫好的作品存為「元件」。而在 3D warehouse 網站中所抓到的模型中，我們為了取得每個元件中

的「線」與「面」資訊，需要檢視「元件」裡面的「線」、「面」資訊。以圖 31 說明，每個元件(元件 1)裡面可能會包著好幾個元件(元件 2-5)，這些被包著的好幾個元件(元件 2-5)，裡面可能又包著好幾個元件，如圖 31 的元件 5 內又包著元件 6、元件 7。當一群「線」、「面」包成元件後，我們便不能隨便取用面與線資訊。

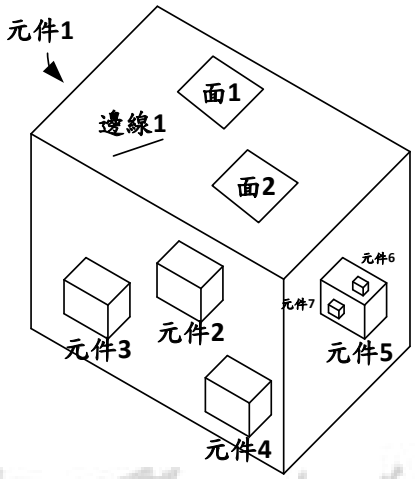


圖 31. 第三方模型的構成示意圖

而如果將所有的元件拆解成許多「線」與「面」，會造成繪圖區的「線」與「面」太多，尤其將複雜「人」、「電視」、「多窗門」...等等元件拆解後，繪圖區的線條變得非常多，而導致頁面變得非常的亂。下一段將介紹不破壞元件的方式分析房子。

圖 32 是將所有的元件實例內的水平面抓出來塗成紅色。過往在分析房子結構時會將元件分解，分解後組不回來，若不用經過拆解就能分析裡面的元素的話，對於破壞結構上面的缺憾即可補足。

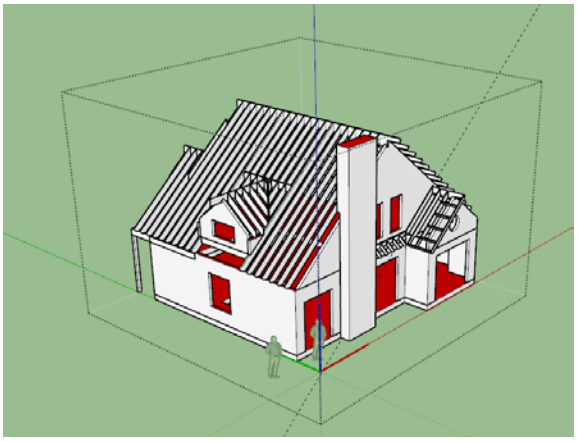


圖 32. 標註所需的面塗為紅色

3.3.3.2 房間資訊

在篩選階段，只要符合模型最底面的水平面，且該水平面面積最大，本論文便會將其視為房間資訊所需水平面。若模型中沒有底面或底面不明確的情況，我們會在模型底下多畫一層底面以方便獲取房間資訊。

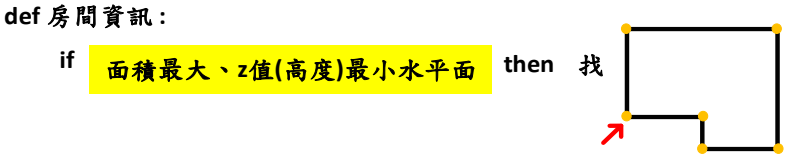
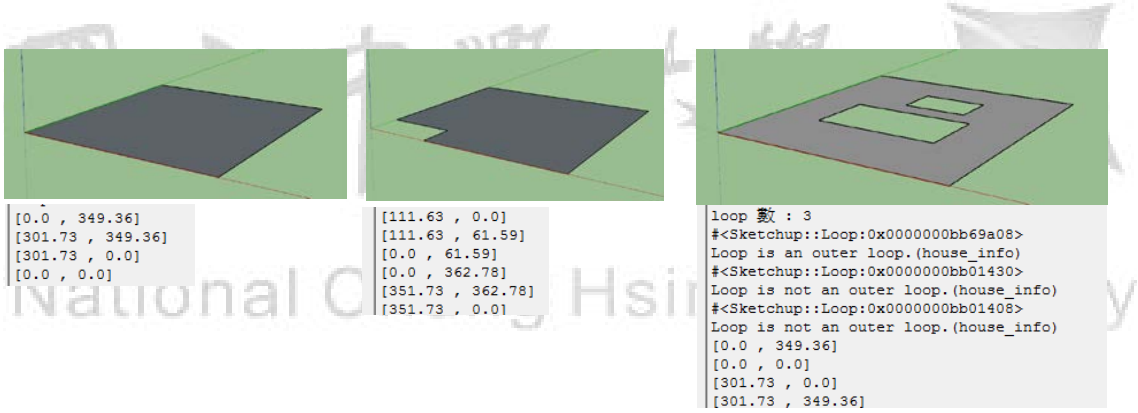


圖 33. 尋找房間資料的示意圖

圖 33 為預想找房間資訊的示意圖，若找到相對應的面，則取面上的頂點，這些頂點必須按照順序排列，以便後續完成平面圖。



(a)四邊形　　(b)六邊形　　(c)四邊形-2 個洞

圖 34. 如何辨別不同情況的水平面

從 3D Warehouse 抓取下來的一些模型，家具跟地面黏在一起，所以取到的水平面還須修正。圖 34 是使用腳本所寫出來印出房間資訊的類別，圖 34 的(a)、(b)分別表示抓取的四邊形及六邊形的座標，而透過圖 35 來解釋圖 34 (c)。圖 35 的四邊形總共有 3 個 loop，最外圍為 Outer loop，而裡面有 2 個 Inner loop。我們不需要 Inner loop，所以將 Outer loop 萃取出來即可。

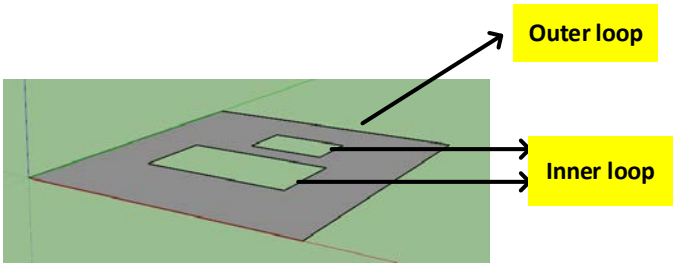


圖 35. 水平面的 Inner Loop 和 Outer Loop

萃取出 Outer loop 以後，因團隊研究需要，取出來的房間資訊要按照逆時針或順時針排列，所以不能夠直接取出「面」上面的「點」。我們必須先抓取「面」上面的「邊」，再取「邊」上的「點」，如此一來，點就會按照順序排列，如圖 36 所示。

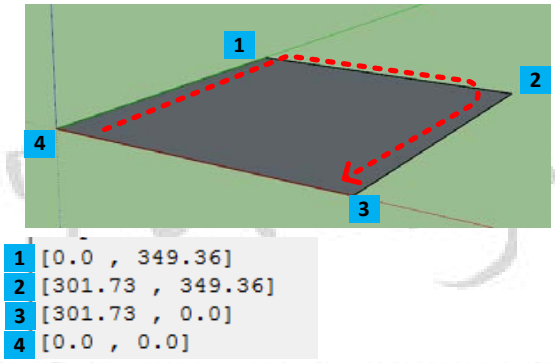


圖 36. 面所屬的頂點排序

本論文在取房間資訊時，以原點為起始點，以逆時針方式排列。在腳本在排列點時，會排出順時針與逆時針兩種情況，而我們統一以逆時針排列。該點若不以原點為起始點，或不以逆時針排列，即會做出修正。

3.3.3.3 門資訊

獲取門資訊較困難，因為一個面上的門可能不只一個，且有些面上會有窗戶...等等的變因造成誤判。圖 37 是找門的判斷式，並取出門上的資訊。

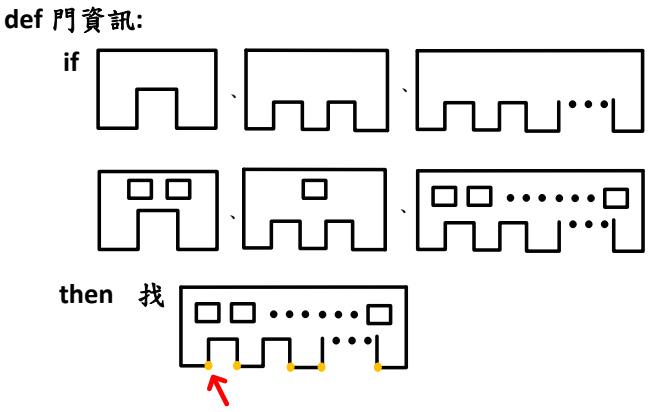


圖 37. 尋找門資料示意圖

不同模型中的門樣式，種類多樣，圖 38 將遇到的分成不同情況介紹。首先遇到圖 38 (a)下載的模型無法抓到門或牆壁，此種情況，本論文會將門與牆壁自行補上去。另一種狀況，如圖 38 (b)所示，是模型中的門位置不適用腳本找門的規則，所以需要人工標註。還有另一種情況為圖 38 (c)，線條過於複雜，可能會造成腳本誤判，若遇到這種狀況，也需要額外處理。

(a) (b) (c)

圖 38. 在上述三種情況中抓取門資訊

如圖 39 所示，模型中也有門上畫門框的種類。為了避免本論文腳本將門框視為門的情況，因此需要更多條件式因應；如圖 40 所示，有些模型中的門為立體，而有些門為平面，因此在撰寫程式上，對於獲取門資訊需仔細考慮。

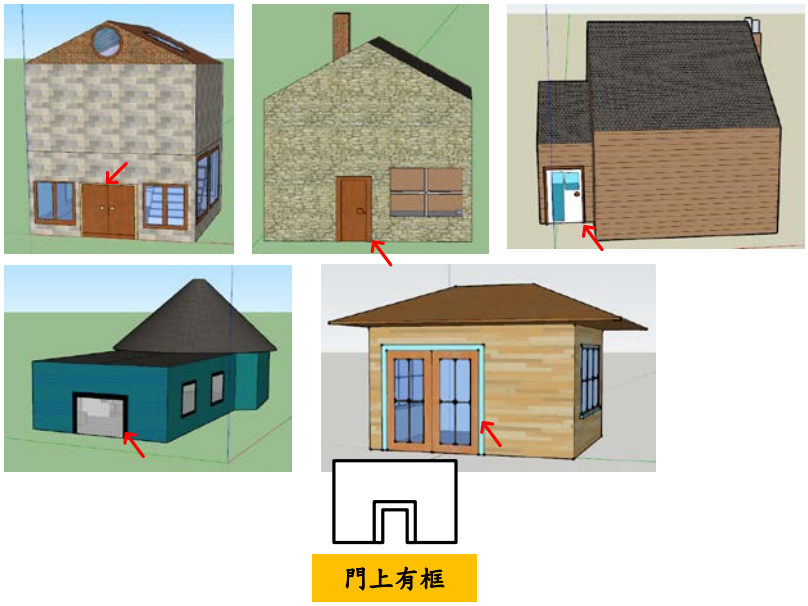


圖 39. 在門上有框的情況抓取門資訊

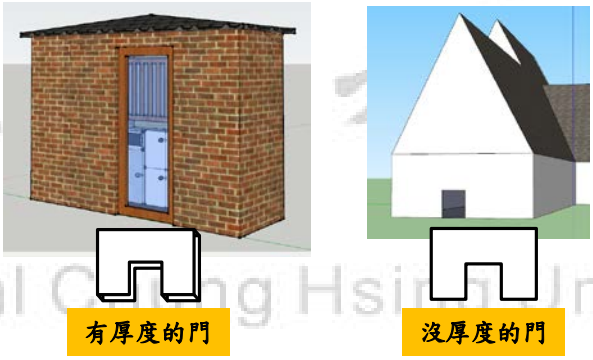
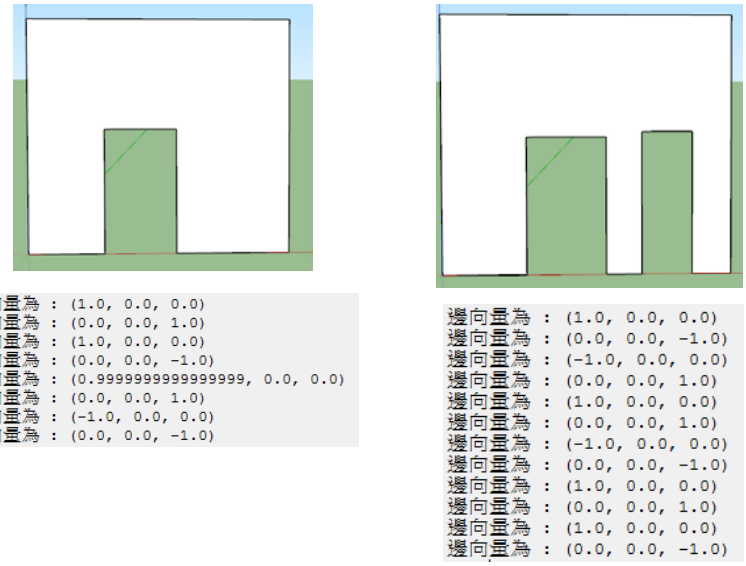


圖 40. 在上述兩種情況中抓取門資訊

因模型中不只有門、地板，還有「人」以及其他的「物件」，這些「物件」都是由許多的點線面所組成，因此若我們只單純的抓取垂直面，會抓取相當多的東西，因此我們需要篩掉我們不需要的東西。

圖 41 整理了 SketchUp 中關於門的特徵，發現當模型只有 1 個門時，有 4 個邊平行 z 軸以及 4 個邊垂直 z 軸，而在 2 個門的情況，有 6 個邊平行 z 軸以及 6 個邊垂直 z 軸。



(a) 1 個門　　(b)2 個門

圖 41. 如何在同個面上抓取不同數量的門

圖 42 是本論文整理出的門資訊的四大特徵。其中特徵 1 為上一段提及的內容，而特徵 2 講述門的底部，至少 4 點共線，多則 6、8、...，只要抓到共線的特徵，即符合特徵 2。

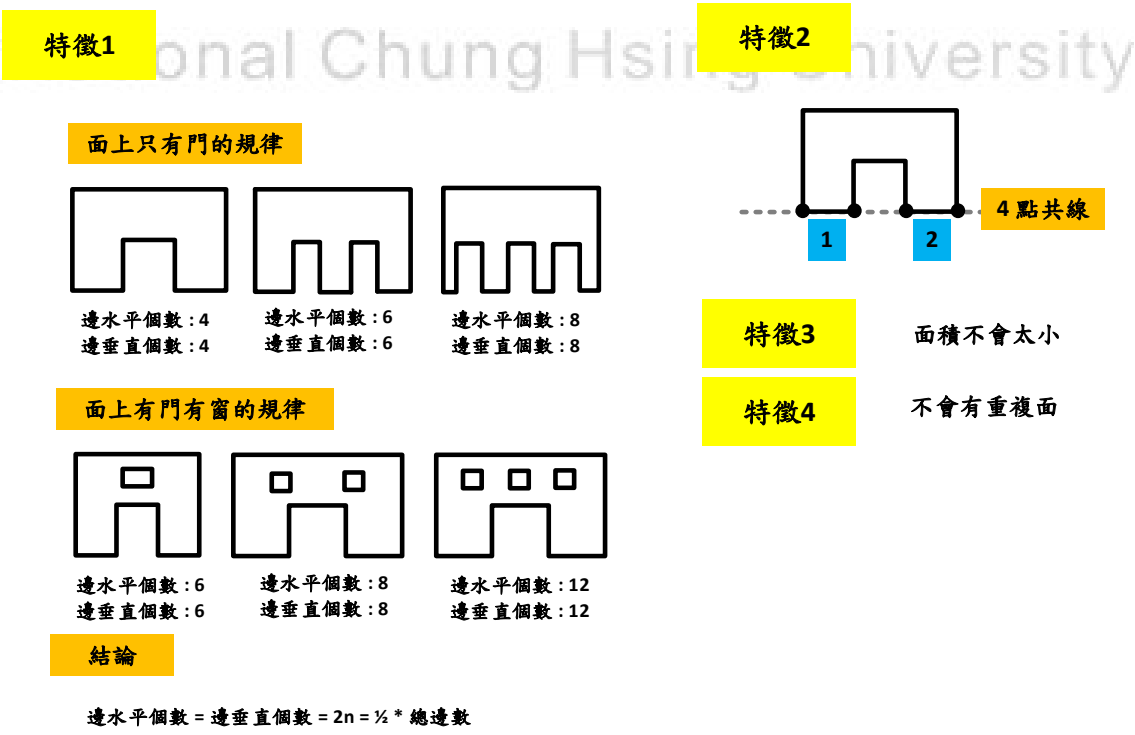


圖 42. 抓取門的篩選條件示意圖

圖 44 說明如何取門資訊。如圖 44 (a)所示，我們當取一個門的面來分析時，不用像房間資訊，需要先取「面」上的「邊」，再取「邊」上的「點」，直接取「面」上面的「點」即可。取非最高和非最低點，即為我們的「門資訊」。除此之外，當紅色點的個數大於 2，便需要排列，如圖 44 (b)所示。

圖 43. 如何抓取同高度的門示意圖

圖 44 是尋找「門資訊」的結果。假設一個面有 6 個門，且門高度都一致的情況下，每一個座標值，依據每個座標值(x+y)的值大小進行排序。

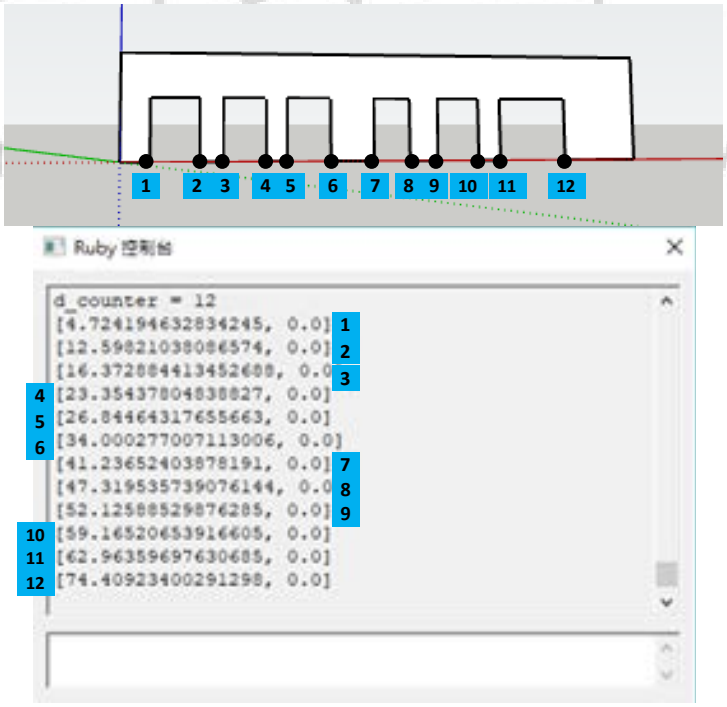


圖 44. 使用 Ruby Script 抓取同高度門的輸出

3.3.3.4 拍照資訊

圖 45 所示，只要水平面為模型的最底面，且該水平面面積最大，本論文便會將其視為想要的水平面。本論文在獲取的面中隨機的設定拍攝點，拍取周圍的環境。拍照的同時也會記錄在拍照位置、拍取的目標物(target)，以及與 x 軸所夾的角度，將這些資訊標註在照片中。

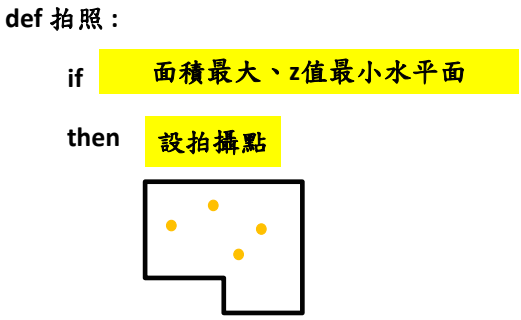


圖 45. 抓取拍照點示意圖

如圖 45 所示，本論文取得的面補成一長方形，如圖 46 所示的灰色虛線，接著以網格方式將點分佈上去。若網格分佈點不在面上，再將點刪除，剩下的點隨機取出 10 個，便完成了我們取拍攝點的動作。

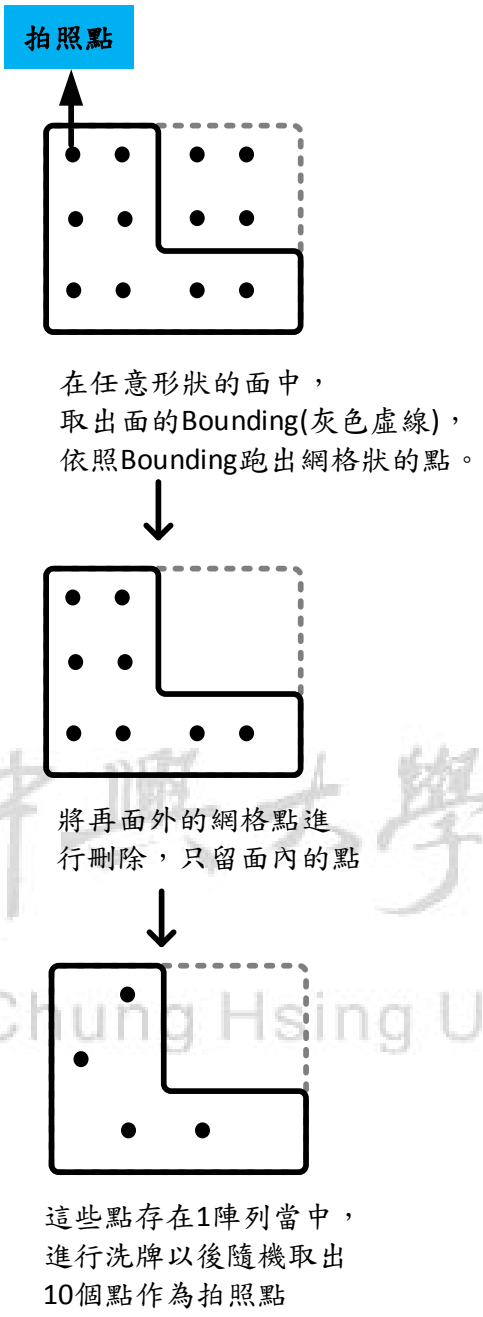
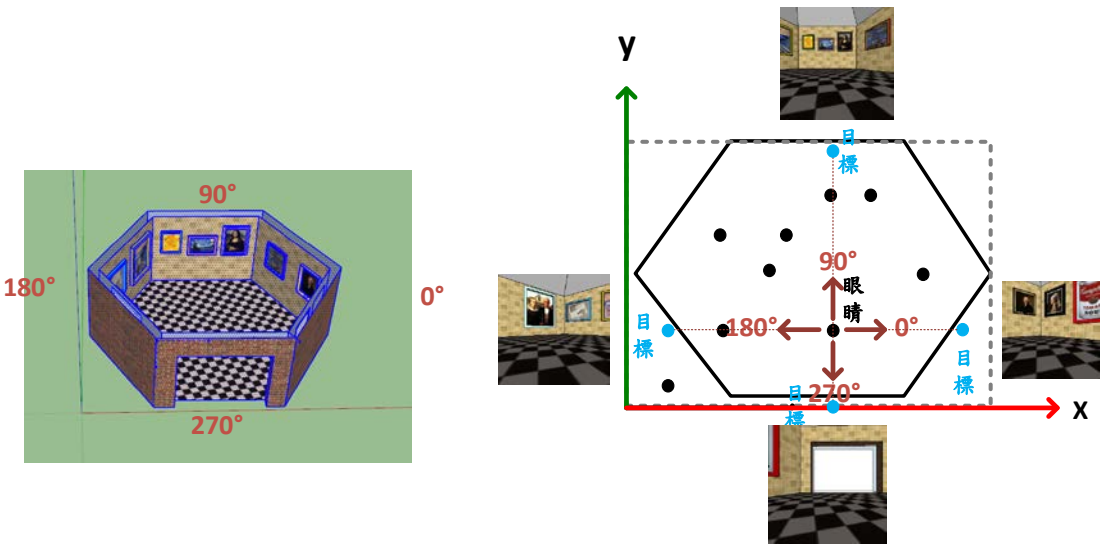


圖 46.設定拍照點示意圖

本論文用如圖 46 取得的拍照點，進行 4 個方位的拍照，如圖 47 所示。如圖 47 (b)，每一個拍照點都會以 0°、90°、180°、270°四個角度進行攝影，所以本論文每一個房間一共會拍出 40 張照片。每一張照片都擁有自己拍照點的座標與和與 x 軸所夾的角度。



(a)俯視圖　　(b)對照圖

圖 47. 使用 Ruby Script 擷取影像的方式

Sketchup 的 Ruby API 中並沒有 0 到 360° 的概念，舉例來說，如果 vect1 = (1,0,0)而 vect2 = (0,1,0)時得到的角度為 90° ；若 vect1 = (1,0,0)而 vect2 = (0,-1,0)時，得到的角度一樣會等於 90°，不會是-90°或 270°。為了修正這個問題，我們會在 vect1 與 vect2 中間畫一個同時垂直於 vect1 與 vect2 的法向量，利用 vect1 與 vect2 的外積，與法向量內積後判斷方向為順時鐘/逆時鐘，若為逆時鐘的情況，direction 的值為正；若順時鐘的情況，direction 的值為負。

預想中，利用 SketchUp API 拍攝照片應全為室內場景，然而實際上，SketchUp API 有些拍下來的照片場景，卻拍取到室外場景，就如圖 48 所示，上述狀況稱為「穿模」。為了解決穿模狀況，我們嘗試使用過 Sketchup API 中的 Raytest 功能，Raytest 可以從拍攝點位置發出給定向量，回傳碰到的點，此點為 SketchUp API 照相機功能的「目標」參數，如此一來，SketchUp 便不會產生異常的狀況。

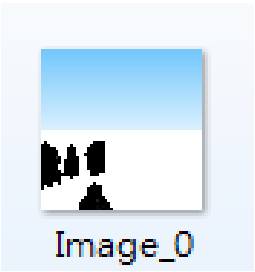


圖 48. 異常情況

Raytest 的功能為，當輸入 1 個點、1 個向量時，會以輸入的「點」作為起始點，給入的「向量」為方向，當撞到第一個物件的時候，回傳碰撞「點」。

然而若起始點，給定的向量為方向，卻找不到碰撞「點」，Raytest 便無法回傳值而出現錯誤。實際上，Ruby 腳本進行拍照時，門處於打開的狀態，便會造成錯誤，所以我們用以下方式解決。

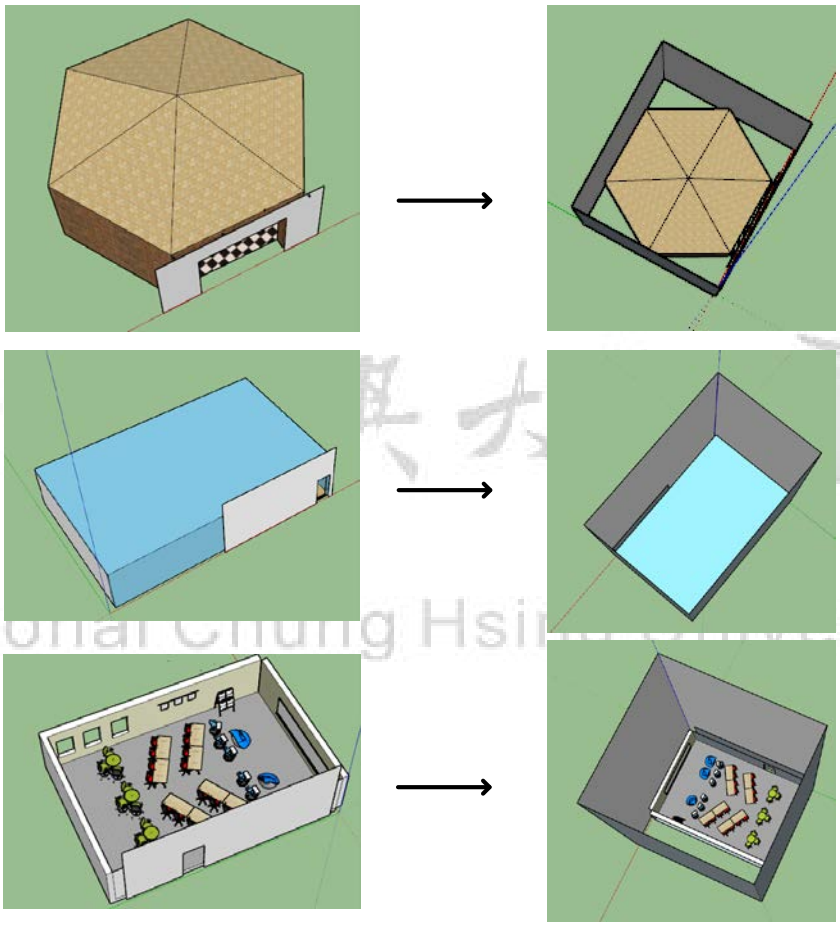


圖 49. 將模型包起來

圖 49 為以一模型舉例說明，我們將房子的四周包起來，使用 Raytest 從眼睛位置發射向量，一定碰得到點，成為目標參數。

3.3.4 After SketchUp

有了房間、門、照片資訊後，我們的輸出結果如圖 51。假設有 1000 棟

的模型，會有對應的 1000 個資料夾，每一個資料夾會有 1 個文件檔及 40 張照片(10 個拍照點*4 個方位)。在文件檔的第一行為房間資訊，房間資訊一共有 50 對座標值，若此房間資訊不足 50 對則後面補 0，而第二行為門資訊，門資訊一共有 10 對座標值，即最多 10 個門，若此門資訊不足 10 對則後面補 0。第三行以後為照片資訊，第三行對應 Image_0.jpg、第四行對應 Image_1.jpg、第五行對應 Image_2.jpg…以此類推。

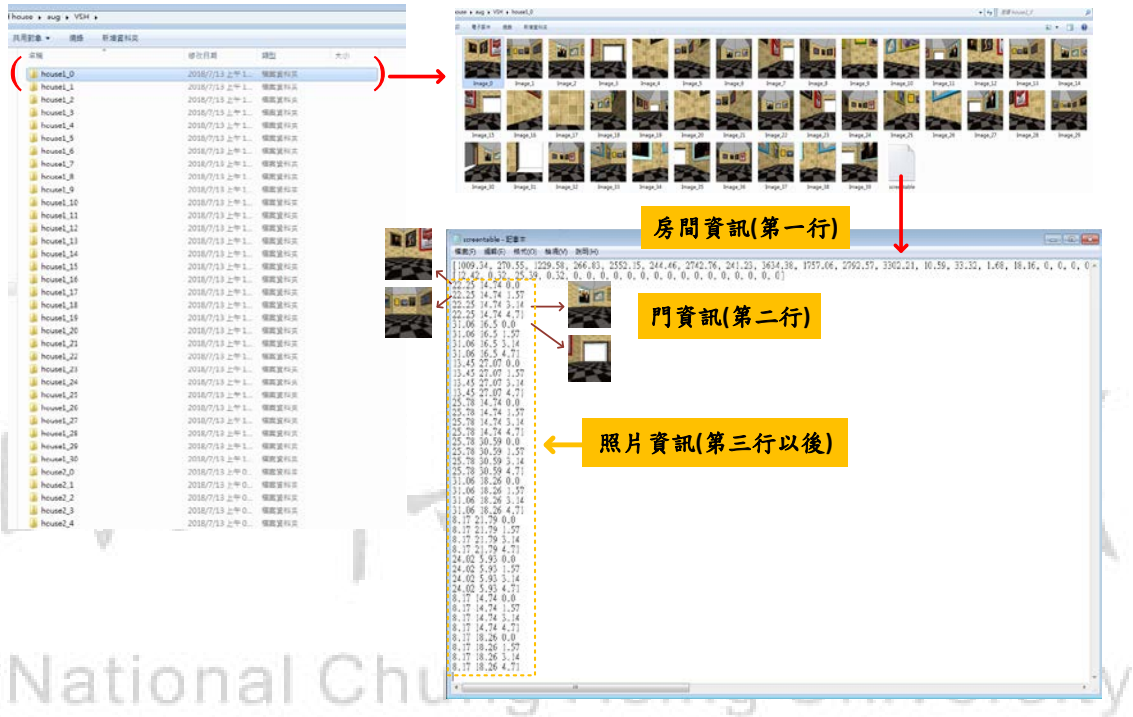


圖 50. 輸出給深度學習的資料

舉一個例子來說明，當要將檔案傳送給深度學習演算法時須先轉成 pickle 檔(.pkl)，亦即將房間/門資訊、照片及照片資訊存成三個矩陣後壓縮、合而為一個資料型態。

Pickle 是 Python 語言的一個標準模組，實現了基本的資料序列化和反序列化。通過 pickle 模組的序列化操作，我們能夠將程式中運行的物件資訊保存到檔案中，而通過 pickle 模組的反序列化操作，我們能夠從檔案中，創建上一次程式保存的物件。

除了 pickle 以外，json、yaml、xml…也是資料序列化的方，
不同於 pickle 的地方在於，開啟 pickle 檔(.pkl)時會以亂碼方式呈現，而 json、yaml 會以 Python 字典方式呈現資料。

圖 51. Pickle 儲存的資料格式

圖 51 中的(4,40,3)代表 4 棟、40 張、3 個幾何值(x,y,angle)；(4,40,56,56,3)代表 4 棟、40 張、56 像素*56 像素的 RGB 照片；(4,120)代表 4 棟、120 代表 60 個(x,y)數值的房間及門資訊。

第四章 實驗結果

4.1 貓狗混合分類及遷移學習實驗結果

在 2.2 節，本論文描述利用不同的真實及虛擬資料的比例訓練演算法，而表 2 為貓狗辨識依照不同的真實及虛擬資料比例所跑出來的結果。此結果的趨勢並不明顯。

表 2. 真實/虛擬資料比較對應準確度

Real / Virtual	200+200	400+400	600+600	800+800	1600+1000

0+0	0.6875	0.6875	0.7500	0.7500	0.7500
200+200	0.7500	0.6875	0.6875	0.8750	0.7467
400+400	0.6875	0.7500	0.9300	0.6875	0.9375
800+800	0.6875	0.8125	0.6875	0.8750	0.6250
1600+1000	0.7500_1	0.5000_2	0.8750	0.6875_3	0.9375

表 3 為利用虛擬貓 1600 張+狗 1000 張訓練出來的權重，利用遷移學習訓練真實貓+狗而跑出來的準確率。虛擬貓+狗訓練出來最終的虛擬貓狗準確度在 0.5625~0.7500 間跳動，而表 3 是以虛擬貓狗權重為初始值，固定前面不同層數與不同真實貓+狗數量跑出來的真實貓狗準確度。真實的貓狗在訓練數目為貓+狗在 200 張+200 張、400 張+400 張、800 張+800 張的狀況下，且初始權重值為隨機時，準確度分別為 0.6875、0.6875、0.7500。可以看出在真實貓狗數量少時，虛擬貓狗有明顯的幫助。

表 3. 遷移學習-利用虛擬貓/狗訓練的權重幫助訓練真實貓/狗

固定層數 / 真實貓+狗	**Non-transfer learning**	**0**	**4**	**7**	**11**	**15**	**17**	**18**
200+200	0.6875	0.7500_1	0.8750	0.8750	0.8125	0.8750	0.8750	0.8125
400+400	0.6875	1.0000_2	0.9375	1.0000	0.6875	0.6250	0.9060	0.7500
800+800	0.7500	0.8125_3	1.0000	0.8750	0.8750	0.8750	0.6250	0.6250

表 4. 遷移學習-利用真實貓/狗訓練的權重幫助訓練虛擬貓/狗

固定層數 / 虛擬貓+狗	**Non-transfer learning**	**0**	**4**	**7**	**11**	**15**	**17**	**18**
200+200	0.5625	0.6250	0.6875	0.5625	0.7500	0.5625	0.5625	0.5625
400+400	0.6875	0.6250	0.8125	0.6250	0.6875	0.7813	0.6250	0.4685
800+800	0.5000	0.7500	0.8125	0.5625	0.8125	0.6875	0.8125	0.4375

表 4 為利用真實貓 1600 張+狗 1000 張訓練出來的權重，利用遷移學習訓練虛擬貓+狗而得的準確率。真實貓+狗訓練出來最終的準確度在 0.9375~1.0000 間跳動，而表 4 是根據固定前面不同層數與不同虛擬貓+狗數量跑出來的準確度。虛擬的貓狗在訓練數目為貓+狗在 200 張+200 張、400 張+400 張、800 張+800 張的狀況下，且初始權重值為隨機時，準確度分別為 0.5625、0.6875、0.5000。看不出在虛擬貓狗數量少時，真實貓狗資訊有明顯的幫助。

在表 4 的 Non-transfer learning 的欄位中出現準確度 0.5000，是因為虛擬的貓+狗照片的特徵因不同作者而相片間差異太大，所以導致最後機器訓練不佳，因此才有這麼低的準確度。除此之外，在表 4 中固定層數為 18 層時，訓練出來結果不好，因此出現準確度小於一半的情況。

我們分別比較表 2 與表 3 的註 1、註 2、註 3 數值。表 2 的註 1 為利用 2600 張(1000 張狗+1600 張貓)虛擬照片及 400 張(200 張狗+200 張貓)真實照片混合訓練，在測試階段預測出來的準確率；表 3 的註 1 為先利用 2600 張(1000 張狗+1600 張貓)虛擬照片訓練模型後，再使用 400 張(200 張狗+200 張貓)真實照片以「2600 張(1000 張狗+1600 張貓)虛擬照片訓練模型」訓練出來的權重為初始值進行訓練後，在測試階段預測的準確度，而表 2 與表 3 中的註 2 數值及註 3 數值也是一樣的道理。

利用表 2 與表 3 的註 1-3 比較，我們可以發現，表 2 註 1-3 的數值較表 3 註 1-3 所對應的數值還要高，代表其實遷移學習的助益在貓狗分類上是大於資料混和的訓練。除此之外，在表 3 註 1 右邊，固定 4、7、11...、18 層訓練出來的結果都比 0.75 還要高；在表 3 註 3 數值的右邊，固定 4、7、11、15 層訓練出來的結果都比 0.8125 還要高，代表遷移學習的效益，超乎資料混和訓練的結果。

表 3 遷移學習-利用虛擬貓/狗訓練的權重幫助訓練真實貓/狗

固定層數 / 真貓/狗	Non-transfer learning	0	4	7	11	15	17	18
200+200	0.6875	0.7500_1	0.8750	0.8750	0.8125	0.8750	0.8750	0.8125
400+400	0.6875	1.0000_2	0.9375	1.0000	0.6875	0.6250	0.9060	0.7500
800+800	0.7500	0.8125_3	1.0000	0.8750	0.8750	0.8750	0.6250	0.6250

表 4 遷移學習-利用真實貓/狗訓練的權重幫助訓練虛擬貓/狗

固定層數 / 虛擬貓+狗	Non-transfer learning	0	4	7	11	15	17	18
200+200	0.5625	0.6250	0.6875	0.5625	0.7500	0.5625	0.5625	0.5625
400+400	0.6875	0.6250	0.8125	0.6250	0.6875	0.7813	0.6250	0.4685
800+800	0.5000	0.7500	0.8125	0.5625	0.8125	0.6875	0.8125	0.4375

圖 52. 比較表 3 與表 4

如圖 52 所示，我們分別比較表 3 的藍色虛線方框與表 4 的綠色實線方框，我們可以發現藍色虛線方框的數值普遍大於綠色實線方框。我們可以推斷，即便虛擬貓+狗訓練出來最終的準確度不高(0.5625~0.7500)，但因虛擬貓狗取到的特徵囊括真實貓狗取到的特徵，所以讓藍色虛線方框內的準確度普遍都比綠色方框都還高。反之，因真實貓狗訓練出來的結果並無法囊括虛擬貓狗，所以造成表 4 中準確率偏低，甚至有小於 0.5 的情況發生。

由此本論文推論，因虛擬資料為不同創作者依照真實貓狗特徵為依據畫出的作品，因此虛擬資料的特徵是藝術家萃取真實資料精華所得，故有明顯的幫助。若我們所抓取的虛擬數據集的量更多，想必會有更好的結果。

在本團隊研究，我們也可以依循貓狗二元辨識為依據來研究，但因 3D 建模模型產出的資料型態與貓狗資料集不同，且本團隊研究為迴歸問題，故仍需要進一步討論。

4.2 SketchUp 虛擬資料輸出結果

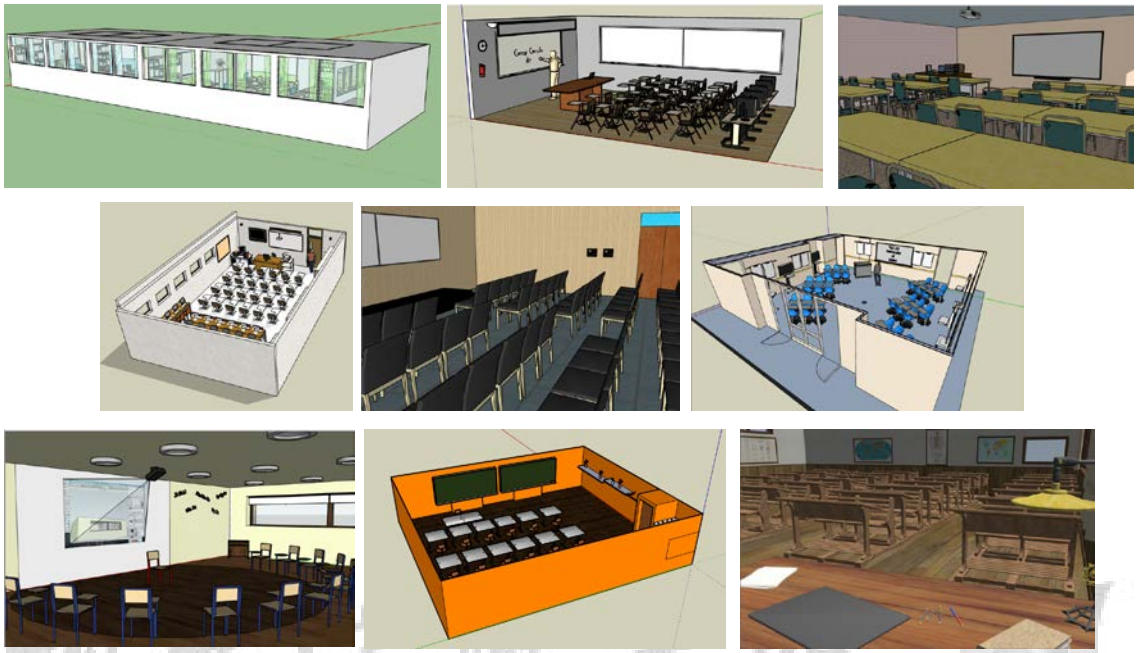


圖 53. 從 3D Warehouse 網站中爬取下來的模型

在 3.3.1 節中提到，由爬蟲爬取下來的圖片由圖 53 所示。這些作品都是由不同作者畫出來的模型，免費提供給其他使用者下載，而作品之間差異性都很大。

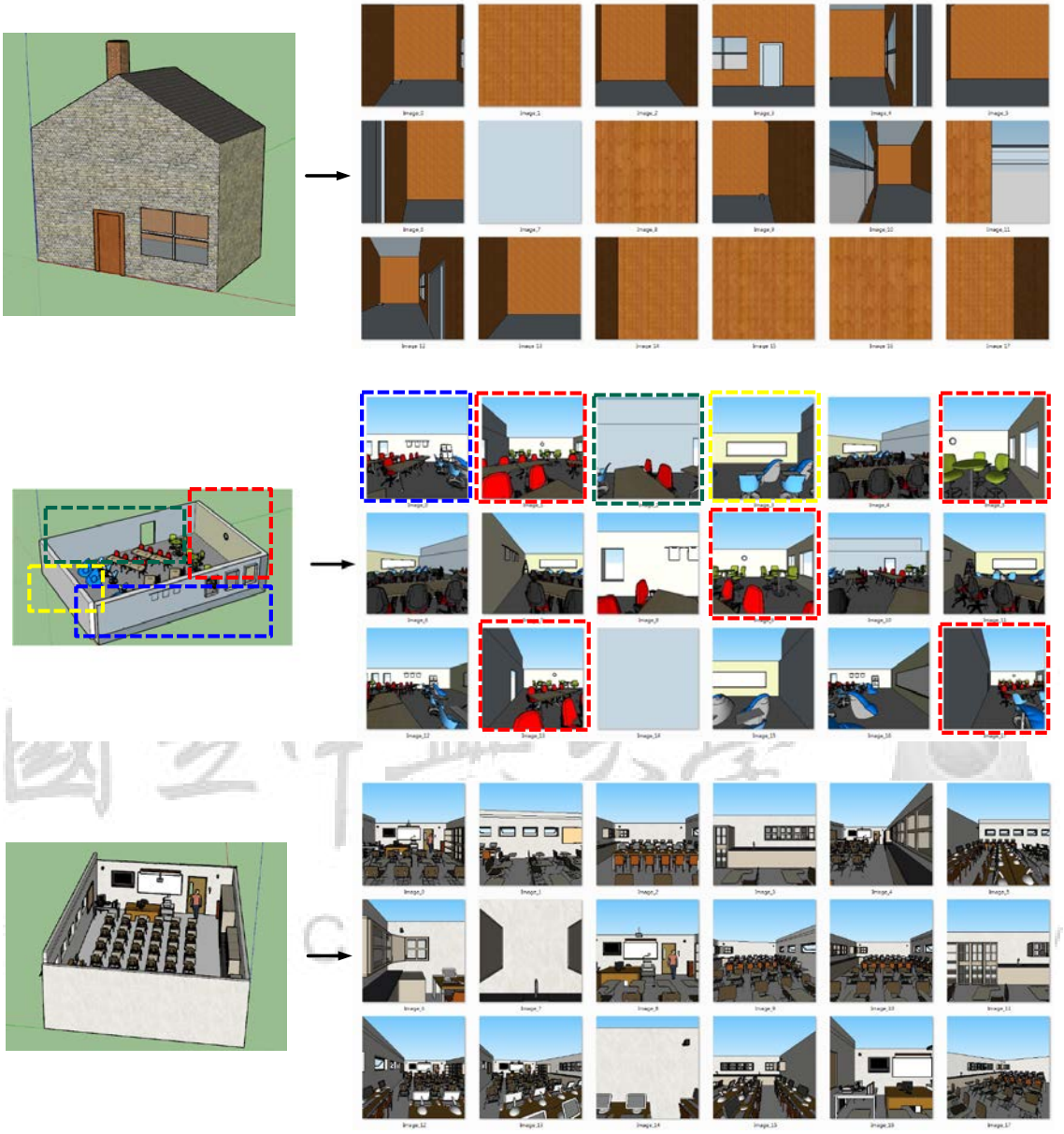


圖 54. 透過 SketchUp 的 Ruby Script 拍下來的照片

在 3.3.3.4 節中所提到我們隨機選取 10 個點，以 4 個方位進行拍攝，一共會有 40 張照片，圖 54 為 3 組房間模型，藉由 Ruby 程式語言腳本拍攝的室內照片，因論文版面的關係，每一組只擷取 18 張。

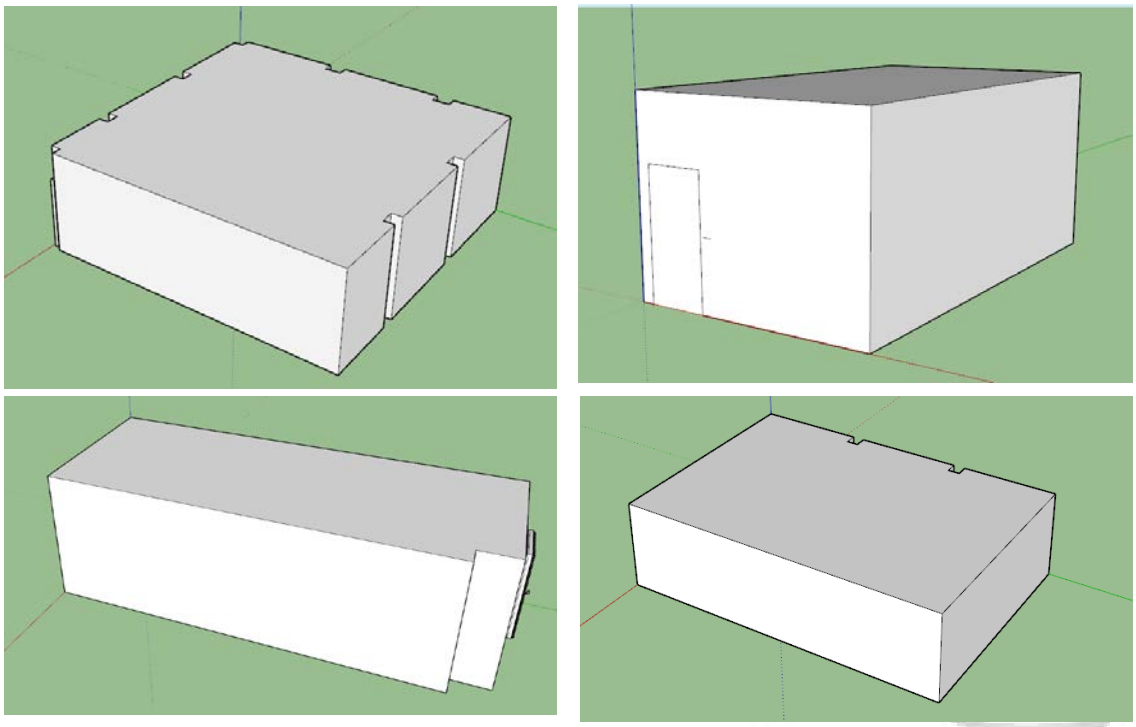


圖 55. 以真實世界為基底所建立之虛擬世界資料

3.2.2 節中提到，本論文會以真實世界中建好的資料為基礎，在 SketchUp 建置相似的模型。模型外觀如圖 55 所示，我們利用真實世界的「房間資訊」及「門資訊」建置模型，建置完模型後，也同樣依照圖 50 的方式，拍攝照片和寫入資訊，而圖 56 是依據 Ruby 程式語言，執行程式後拍下來的照片。

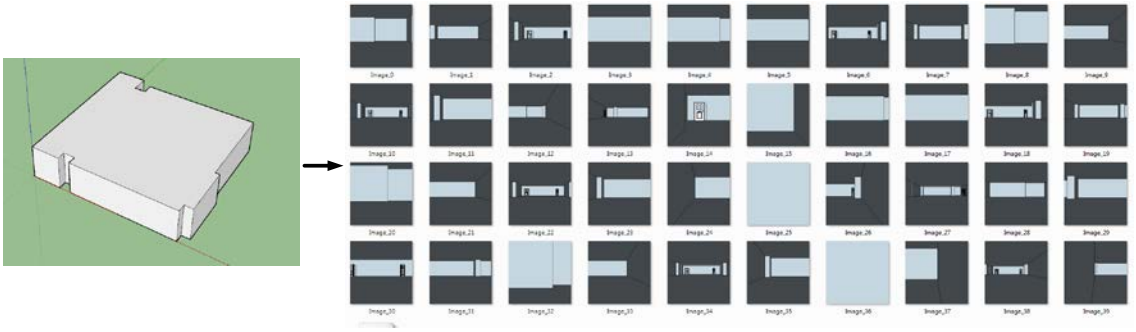
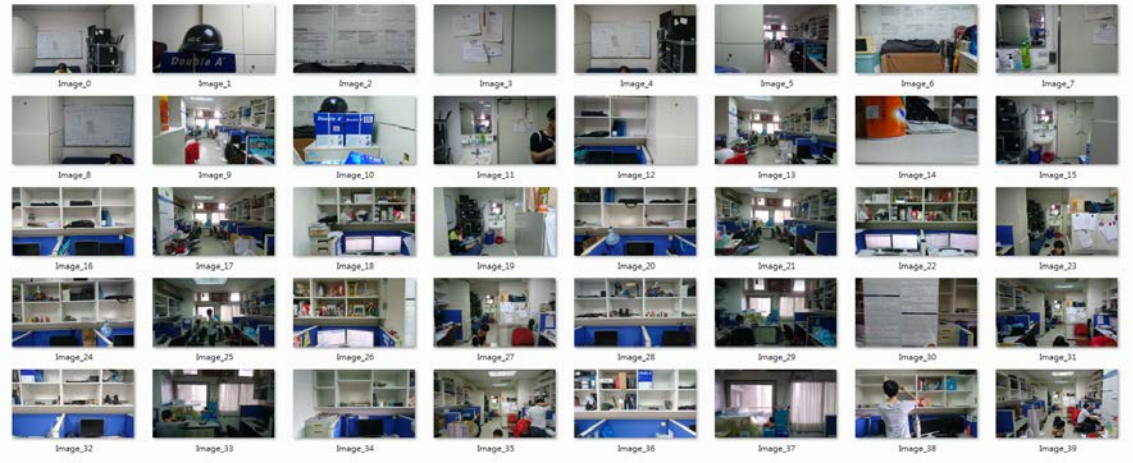


圖 56. 透過 SketchUp 的 Ruby Script 拍下來的照片

4.3 真實世界資料

實驗室715



陽台

圖 57. 真實世界照片

在 3.2.1 章節介紹的真實世界的資料[3]，呈現在圖 57 中。圖 57 是以中興大學電機大樓 715 及某建築物陽台作為例子，我們在室內隨機選取 10 個點作為拍攝點，每一個拍攝點以前、後、左、右四個方位進行拍攝，一共 40 張照片。

4.4 數據集處理結果

```
info_of_house_and_doors : (4499, 120)
imgDatass               : (4499, 40, 56, 56, 3)
geo_infoss              : (4499, 40, 3)
 Xgeo, Ximg, Y :  (4499, 40, 3) (4499, 40, 56, 56, 3) (4499, 120)
```

```
info_of_house_and_doors : (4499, 24)
imgDatass               : (4499, 40, 56, 56, 3)
geo_infoss              : (4499, 40, 3)
 Xgeo, Ximg, Y :  (4499, 40, 3) (4499, 40, 56, 56, 3) (4499, 24)
```

圖 58. 深度學習演算法輸入資料格式

在 3.3.4 節中，本論文提到將 SketchUp 產生出來的虛擬資料轉換成深度學習演算法讀取的資料形式，圖 58 是將虛擬資料轉換成 pickle 檔案的最後資料格式，而圖 58 中的 4499 為虛擬資料的筆數，40 為每一筆資料中的照片個數。

在 2.3 節提到，本研究的平面圖是由大量的座標點所連成，所以實際上深度學習演算法的真正輸出為座標點。圖 58 上圖與下圖中的 120 與 24，分別代表不同版本的「房間資訊」、「門資訊」的集合。120 為 100 個數值的房間資訊與 20 個數值的門資訊，24 為 8 個數值的房間資訊與 16 個數值的門資訊。會有 2 個版本的差異，是因為深度學習對於太複雜的室內架構辨別困難，且室內架構中的細節，如突起的牆角及微小型的側柱，就算忽略了也不會影響我們對於室內空間的認知，因此才有 24 個數值的版本。

4.5 建立虛擬數據集的效益

本論文在爬蟲階段，爬取了 150,000 筆的虛擬資料，包含了利用”house”關鍵字下載下來的 100,000 筆模型，以及利用”classroom”、”bungalow”等關鍵字下載的 50,000 筆模型。經過初步篩選後，選出了 1,000 筆可用模型。透過這 1,000 筆，筆者整理出規則後，寫在 SketchUp 的 Ruby API 腳本中。執行腳本後，這 1,000 筆中，僅 200 筆模型符合本論文的腳本規則，其餘 800 筆皆產生異常。本論文利用 200 筆的模型，進行房間的拉大拉小及家具的替換後，擴增成 2,000 筆資料提供深度學習演算法進行訓練。

因為演算法的關係，目前可用的資料筆數占全部資料筆數的 0.15%。若能夠用更有系統，或更好的方式找到房間資訊與門資訊，能夠使用的資料筆數會更高。

4.6 團隊研究結果

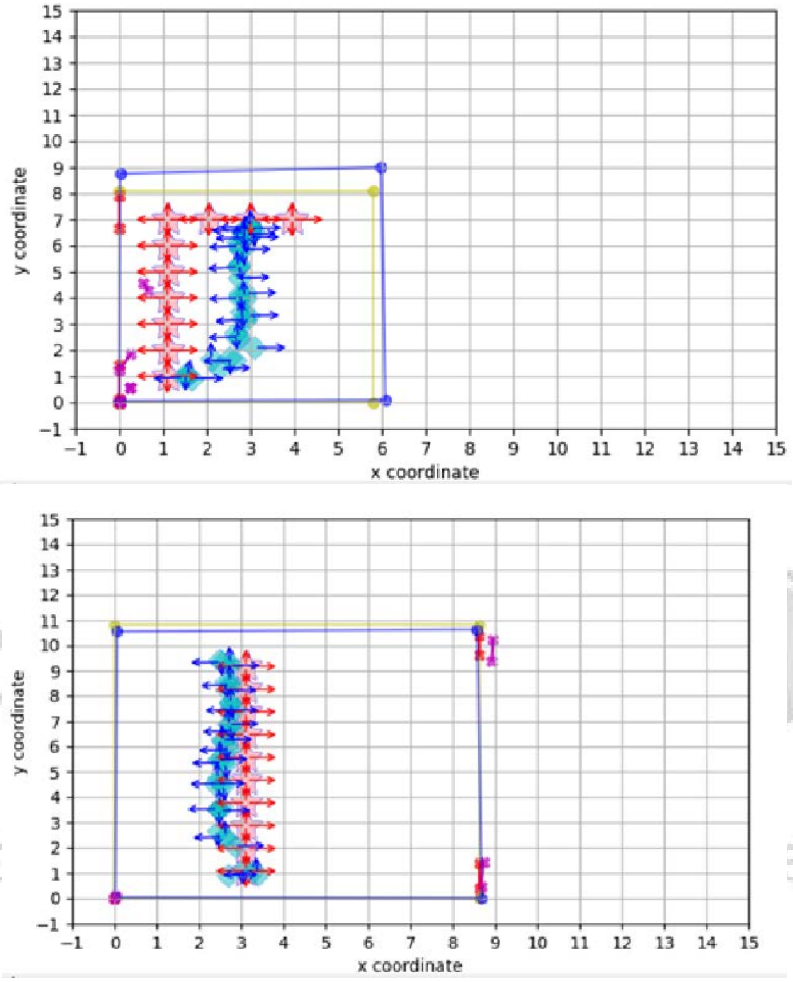


圖 59. 預測結果和真實結果的比對

在 3.2 章中，我們介紹了團隊研究目的，及深度學習架構，而圖 59 正是研究架構中的最後結果[2]。圖 59 中的黃色長方形為真實房間大小，而藍色長方形為預測出來的資料，此外，圖 59 的上圖左側及下圖右側的兩紅色點連起來的紅色線為真實門位置，而圖 59 的上圖左側及下圖右側的兩粉紅色點連起來的粉紅色線為預測的門位置。長方形中間的紅色點為真實的拍照點位置，而長方形中間的藍色點為預測的拍照點位置。

然而本論文由 SketchUp 所提供的虛擬資料，對團隊研究無明顯幫助，細節請參考[2]。

第五章 結論與未來展望

5.1 結論

本論文著重於深度學習中數據集的探討。本團隊研究的數據集由真實世界及虛擬世界所組成，在貓狗的二元辨識實驗中，我們發現虛擬資料可幫助真實資料。

團隊研究的虛擬資料主要由本論文的 SketchUp 產生。作者也在 3D Warehouse 網站上爬取模型。由於 SketchUp 的 Ruby API 提供了健全且方便的功能，本論文能夠順利地將大部分的手續程式化，使電腦自動產生我們所需要的資料，再利用這些資料訓練深度學習演算法。

5.2 未來展望

本團隊研究目標為擴大 Google map 的室內地圖應用，若我們將團隊研究演算法放於伺服端，當所有使用者使用本研究的演算法畫平面圖，這些資料能幫助我們越來越精準的預測出室內地圖。然而因人力及時間不足的關係，並沒有涉足雲端技術的部份，未來要如何進入雲端，是值得探討的議題。

SketchUp 所提供的虛擬資料，對團隊研究無明顯幫助，是目前團隊最想解決的問題，這應該是 SketchUp 提供訓練的虛擬資料不夠逼真。在[11]中提及的 UnrealCV，以及 Blender、UE4 等軟體的渲染技術，都已經發展成熟。如果能將渲染技術程式化並應用，將能大大的改善提供訓練的虛擬資料，進而對團隊研究有明顯的幫助。

參考文獻

[1] 李正東,"當 SketchUp 遇見 Ruby－邁向程式化建模之路" , July 2015.

[2] 鍾隆揚,"基於深度學習之室內定位與地圖繪製" , Nov. 2018, 中興大學碩士論文

[3] 薛至辰," 基於手機平台來取得室內空間尺度與影像資訊以建立深度學習室內定位/地圖繪製之資料庫" , Dec. 2018, 中興大學碩士論文

[4] Yeh James, " https://medium.com/@yehjames/資料分析-機器學習-第 5-4 講-機器學習進階實用技巧-正規化-8dd14fcd3140", Dec. 2017

[5] 李宏毅 , " https://www.youtube.com/watch?v=qD6iD4TFsdQ&t=156s ", Dec. 2016

[6] Iamya, HTML DOM, " https://ithelp.ithome.com.tw/articles/10108783 " , Nov. 2012

[7] 莫凡, " https://morvanzhou.github.io/tutorials/python-basic/threading " , Nov. 2016

[8] M. Johnson-Roberson, C. Barto, R. Mehta, S. Nittur Sridhar, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" , Jun. 2017.

[9] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, andA. M. Lopez, "The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes" , In Proc. CVPR , 2016

[10] W. Qiu and A. Yuille. Unrealcv, " Connecting computer vision to unreal engine." ,2016.

[11] Weichao Qiu, Fangwei Zhong , " Unrealcv Virtual worlds for computer vision" , 2017

[12] Dishashree Gupta , " Transfer learning & The art of using Pre-trained Models in Deep Learning ", Jun. 2017

[13] Sebastian Ruder," Transfer Learning - Machine Learning's Next Frontier ", Mar. 2017

[14] kennethreitz, "http://html.Python-requests.org " update at Mar. 2018

[15] w3schools.com , " https://www.w3schools.com/js/js_ajax_intro.asp "

[16] eztrust , "https://www.eztrust.com.tw/html/faq/qa_show.aspx?id=47 "

[17] Hardik Vasa, Google-images-download, from " https://github.com/hardikvasa "

[18] Transfer Learning using Keras and VGG ,

“ https://riptutorial.com/keras/example/32608/transfer-learning-using-keras-and-vgg ”
[19] Instagram API, “https://developers.facebook.com/docs/instagram-api/v2.11”