中華民國第 59 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

團隊合作獎

052502

智慧門禁─結合影像辨識與 IoT 之應用

學校名稱：臺中市立臺中第一高級中等學校

作者：	指導老師：
高二 黃宥翔 高二 陳淳益	柳佩君

關鍵詞：物聯網、影像辨識、Android Studio

摘要

本研究觀察到現有門禁的流程仍有改善空間，因此著手製作與傳統門禁不一樣的系統，希望能改善其繁複的使用流程。透過物聯網、深度學習和手機應用程式的組合，便能實現這樣的效果。在門前透過樹莓派拍下訪客的照片，傳到電腦後進行影像辨識，初步決定是否要為其開關門，並將結果傳給主人手機，再由主人決定最終的開關門。本次研究目前已成功造出一套完整系統，能即時傳達通知，且辨識的準確率已可達八至九成。

壹、 研究動機

傳統的門禁系統的流程十分繁瑣，訪客通常得要先按門鈴，然後在門前等候主人來應門，等到主人匆忙趕到門後，還得透過門眼察看；又或者，臨時到訪的訪客到了家門前，卻發現主人不在家而得打通電話協調，種種的現象都顯示傳統的門禁系統仍有很大的改良空間。

因此，本研究打算透過課堂所學之 Python 語言及物聯網之觀念，融合現代的技術，如深度學習、APP 之製作等，製作出一套門禁系統，能夠取代傳統門禁，簡化其流程，讓使用者在使用上不但覺得方便，又不失門禁的安全性。

貳、 研究目的

本研究希望設計一套系統，透過物聯網及人工智慧將訪客到來的消息，快速且直接的傳至主人的手機，將上述繁複的過程簡化，實做出能改善生活問題之門禁系統。

期待能將現代發達的科技予以實踐，活用課餘所學，探索方便現代人類生活的另一種可行性。

參、 研究設備

一、硬體設備

(一) 筆電

1. 型號:ASUS UX430UQ

(1) 處理器:Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (4 CPUs), ~2.9GHz

(2) 作業系統:Windows 10 家用版 64-bit

2. 型號: ACER Nitro AN515-51

(1) 處理器: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz (4 CPUs), ~2.5GHz

(2) 作業系統: Windows 10 家用版 64-bit

(二) Raspberry Pi 3 樹莓派

(三) IR sensor module 紅外線感應模組

(四) Raspberry Pi Camera NoIR 攝影機

(五) 伺服馬達 MG946R

(六) TWM Amazing X3 手機

二、軟體設備

(一) Android Studio

(二) Sublime Text

(三) VNC Viewer

(四) Python 3.5 (64-bit)

(五) 深度學習之函式庫:tensorflow1.9.0, keras2.2.4

肆、　研究過程

一、流程圖

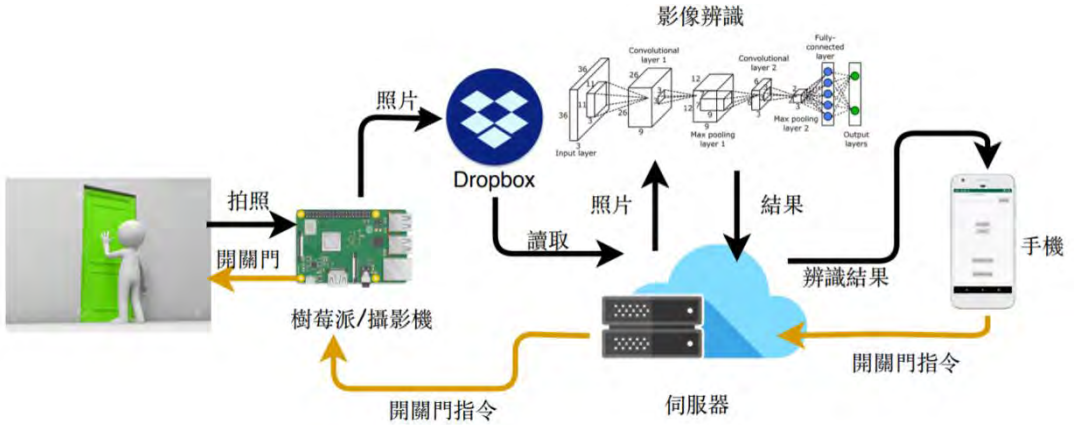


圖 1　系統完整流程圖

二、輸入端----門眼

由樹莓派連接各感應器組成，形成類似門眼的功能，以下將詳述樹莓派的配置與程式。

(一) 樹莓派

1. 操控樹莓派

對於樹莓派，本研究採遠端控制的方式操控和監視它的表現，藉由 putty 和 VNC viewer 來實現。Putty，用來進入樹莓派的終端機，並初始 VNC viewer；VNC viewer，負責登入樹莓派的操作介面。如圖 3，在取得樹莓派當前 IP 後，由 putty 連接上，輸入完密碼後登入，並在終端機中輸入”tightvncserver”如圖 4；接下來開啟 VNC viewer，於 IP 後加上:1 如圖 5；登入後輸入密碼便能直接看到樹莓派的介面，如圖 6。

圖 3　樹莓派 IP 位址

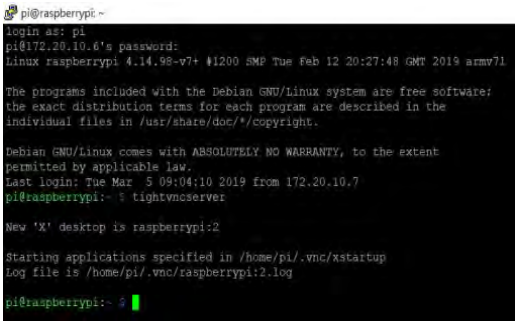


圖 4 putty 介面

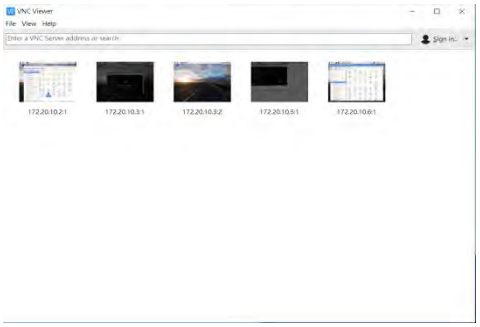


圖 5 VNC viewer 登入介面

圖 6 VNC viewer 樹梅派介面

2. 線路及配置

在樹莓派上，外加一個紅外線感測器，一顆伺服馬達，和一台攝影機 Raspberry Pi Camera(以下簡稱 Pi Camera)。紅外線感測的 INPUT 腳位接於樹莓派的第 8 腳位(GPIO pin 14)，伺服馬達的 OUTPUT 腳位接於樹莓派的第 11 腳位(GPIO pin 17)。Pi Camera 為樹莓派公司自家生產的攝影機，有特定的位置可加裝。

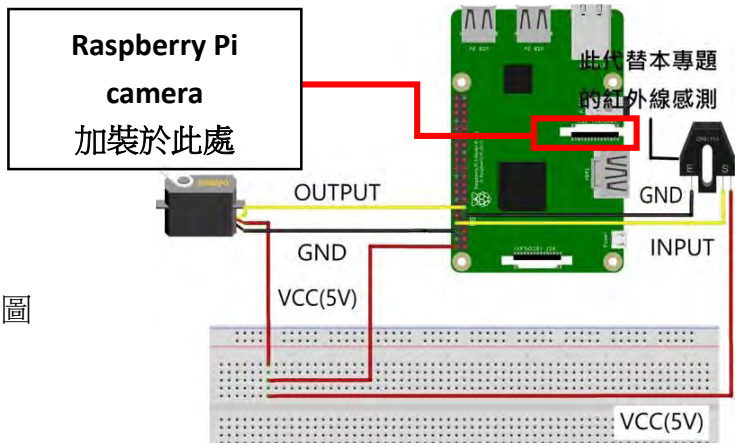


圖 7 樹莓派線路配置圖

表 1 門眼硬體介紹

硬體名稱	型號	功能
樹莓派	Raspberry Pi 3 Model B	執行所有程式之處
紅外線感測器	Rs 55 IR sensor module	感測訪客的到來
Pi Camera 攝影機	Raspberry Pi NoIR Camera V2 800 萬像素/1080p30	拍攝訪客照片
伺服馬達	Servo Motor MG946R	模擬門的開關

(二) 樹莓派程式

樹莓派上有兩支 Python 程式：Initial.py 和 Final.py，最後在執行時，會兩支程式一併執行。Initial.py 負責最一開始的感應和拍照；Final.py 負責收取伺服端(也就是 Server.py，為一支在電腦上運行的程式，將於第三點提到) 傳的訊息，並執行開關門。此部分將著重於介紹 Initial.py，而 Final.py 將於第六點做介紹。Initial.py 程式架構簡單來說就是兩個條件敘述(if)。若有紅外線感測器有感應到人，則執行拍照和上傳一份文字檔至 Dropbox，說明門前有人(文字檔確切內容為“yes”)；若沒有感測到人則只上傳一份文字檔(後以 anybody.txt 代稱)至

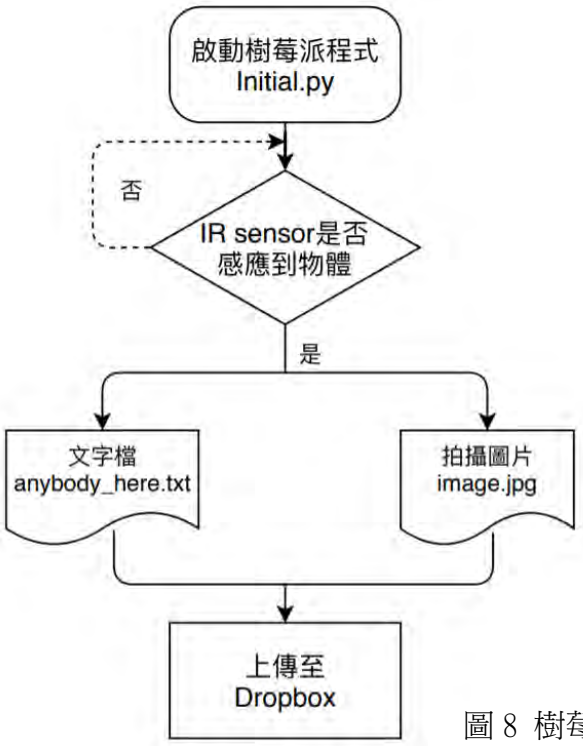


圖 8 樹莓派運行流程圖

Dropbox 說明門前無人(文字檔確切內容為“no”)，anybody.txt 功用為顯示門前是否有人，將在第三點進一步說明。

三、電腦端

電腦在本研究中扮演的角色為連接門眼及使用者的訊息傳送者。

在準備的電腦中會不斷運行兩支程式：toServer 和 Server，如表 1

表 2 電腦運行之程式介紹表

程式	功能
toServer.py	不斷下載文字檔並判斷其內容來決定是否下載圖檔並進行辨識，後傳送結果及圖片至 Server.py
Server.py	將從 Server.py 傳來的訊息及圖檔經過判斷候傳至手機 APP

如此一來，電腦便能將使用者想知道的門前的資訊傳至他們的手機，達到傳遞資訊的效果。

(一) toServer----影像辨識與伺服端之橋梁

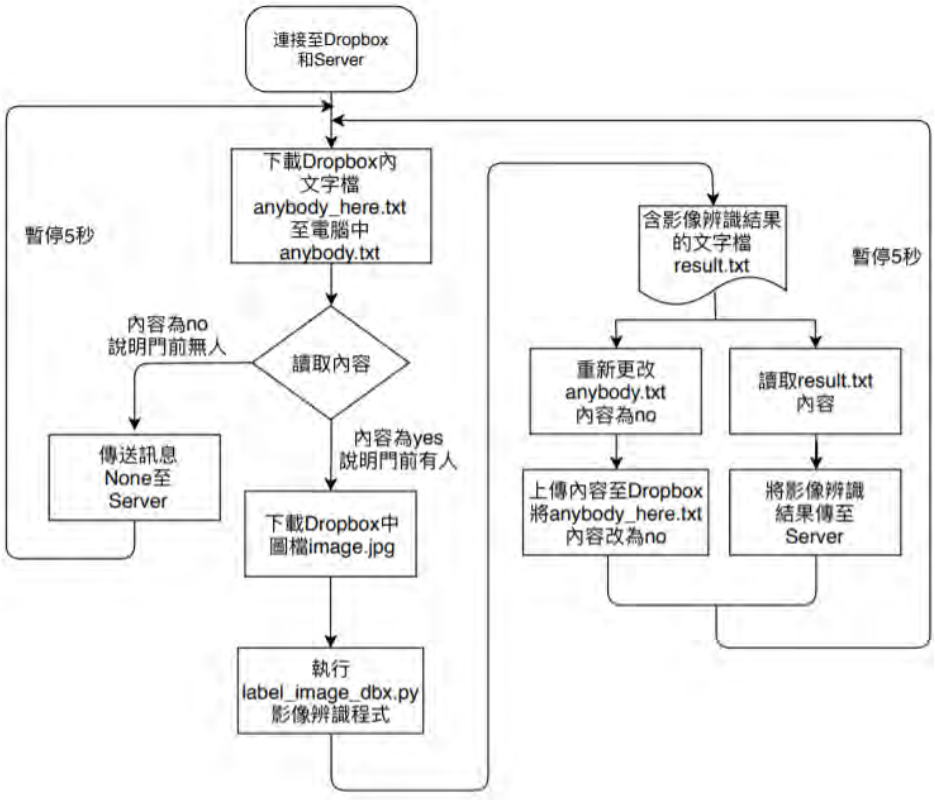


圖 9 toServer.py 程式流程圖

toServer 的主要目的為辨識樹莓派擷取的門前人物為何，並將結果傳至 Server。結構可分為三大部分：收取 Dropbox 資料與連結伺服端(Server)、執行影像辨識的

程式，和傳送結果至 Server。

首先，利用 Websocket 建立其與伺服端(Server.py)的連結。

之後，程式會固定時間跑一次迴圈，透過 Dropbox 提供的函式，下載 anybody.txt 至電腦中，並且，由 anybody.txt 的內容決定是否該進行接下來的辨識過程。

若內容為有偵測到人(anybody.txt 內容為“yes”)，表示 Dropbox 上的影像已經更新，為目前門前偵測到的訪客影像，就從 Dropbox 下載該圖片並執行影像辨識之程式(於第四點作介紹)，並將其辨識之結果傳送到 Server；反之(文字檔內容為“No”)，則門前無訪客，傳送字串”None”至伺服端，經過五秒後再去 Dropbox 收取 anybody.txt，完成一個迴圈。

另外，完成辨識後，將覆寫原 anybody.txt，將內容改為“No”且停止五秒後再運行此迴圈，以免持續傳送相同的結果至使用者手上，而導致其認為不斷有訪客出現在門前。

傳送圖片時，本研究採用將圖片將其 encode 成一 Base64 字串的方式，經由 Websocket 傳送至 Server 再到手機程式，再 decode 回原始圖片。Base64 為一種表示二進位資料的方式，將二進位表示的數字，每六個一組對應其代表的可列印字元進行轉換。

圖 10 圖片 encode 至 Base64 示意圖

(二) Server----訊息傳遞之中樞

此程式收到客戶端(toServer.py)傳來的訊息(感應、辨識結果及圖像)後，首先判斷字串的長度，當字串長度較長時(此處設定為大於 100)，便認定其為圖片進行 encode 後的字串，將其傳至手機 APP；反之，則訊息內容為辨識後的結果，傳送至手機 APP。

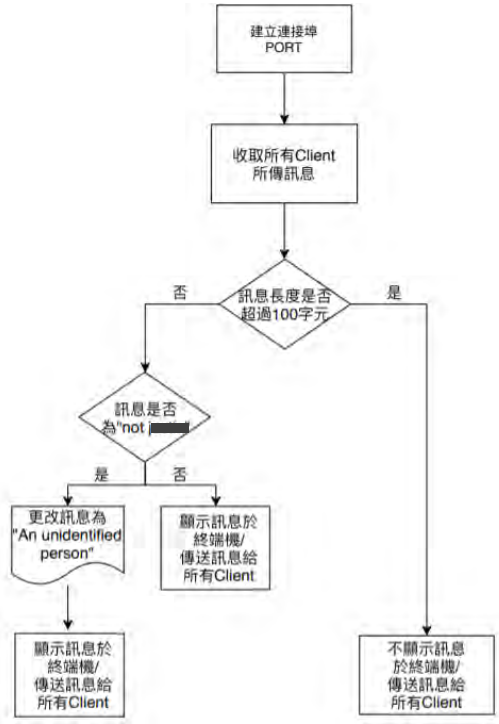


圖 11 Server.py 程式流程圖

四、辨識----移轉學習(transfer learning)

為了增加此裝置的方便和可靠性，本研究加上了影像辨識功能，如此一來，使用者便能不用察看就知道某些預設對象的到來了，藉此也能輔助使用者的判斷。

起初，我們嘗試完整訓練一個模型，但一個模型的參數量實在太大，對電腦是一大負擔，而且也沒有足夠數量的照片，以至於無法完好的訓練整個模型。

經過試用多種訓練方式後，最後選擇使用了「移轉學習」。移轉學習有耗時短且資料需求低的優點，可在有限資源下得到最好的成果。

移轉學習是透過之前其他人重複訓練過的部分模型，來汲取出圖像的特徵，如此一來，需要做的便只有訓練最後的全連接層而已，訓練的時間便會大幅減少，並且能得到理想的準確率。

(一)準備資料集：有策略地蒐集照片-----兼採正臉、側臉及有表情之照片

(二)預處理資料集:利用程式擷取臉部影像並對隨機圖片調整明暗

1. 利用 dlib 的函式擷取臉部的影像

```
DATADIR_TRAIN="C:/Users/Justin/tech_project/cropped_imgs/"
CATEGORIES=["Eagle"]
for category in CATEGORIES:
    i=0
    path = os.path.join(DATADIR_TRAIN,category)
    class_num = CATEGORIES.index(category)
    for img in os.listdir(path):
            print (img)
            img = cv2.imread(os.path.join(path,img))
            img = imutils.resize(img, width=1280)
            shape=img.shape
            detector = dlib.get_frontal_face_detector()
            face_rects = detector(img, 0)
            #x1=0
            #y1=0
            #x2=shape[1]
            #y2=shape[0]
            for i, d in enumerate(face_rects):
              x1 = max(d.left(),0)
              y1 = max(d.top(),0)
              x2 = max(d.right(),0)
              y2 = max(d.bottom(),0)

            print(shape)
            crop_img = img[y1:y2, x1:x2,[2,1,0]]
            print(x1,y1,x2,y2)
            im = Image.fromarray(crop_img)
            im.save("C:/Users/Justin/tech_project/cropped_imgs/"
                +category+"/"+category+str(random.randint(0,200))+".jpeg")
            i=i+1
```

2. 利用 Python 的 PIL 函式庫隨機的調整光暗

(三)開始訓練:本次研究使用的模型為 CNN 的 GooLeNet 的 InceptionV3 模型。訓練的流程大致如下：

1. 下載 InceptionV3 的計算圖，計算圖是 Tensorflow 用來表示計算的形式，一個節點表示一個計算，節點間的鏈結表示計算間的依賴關係。

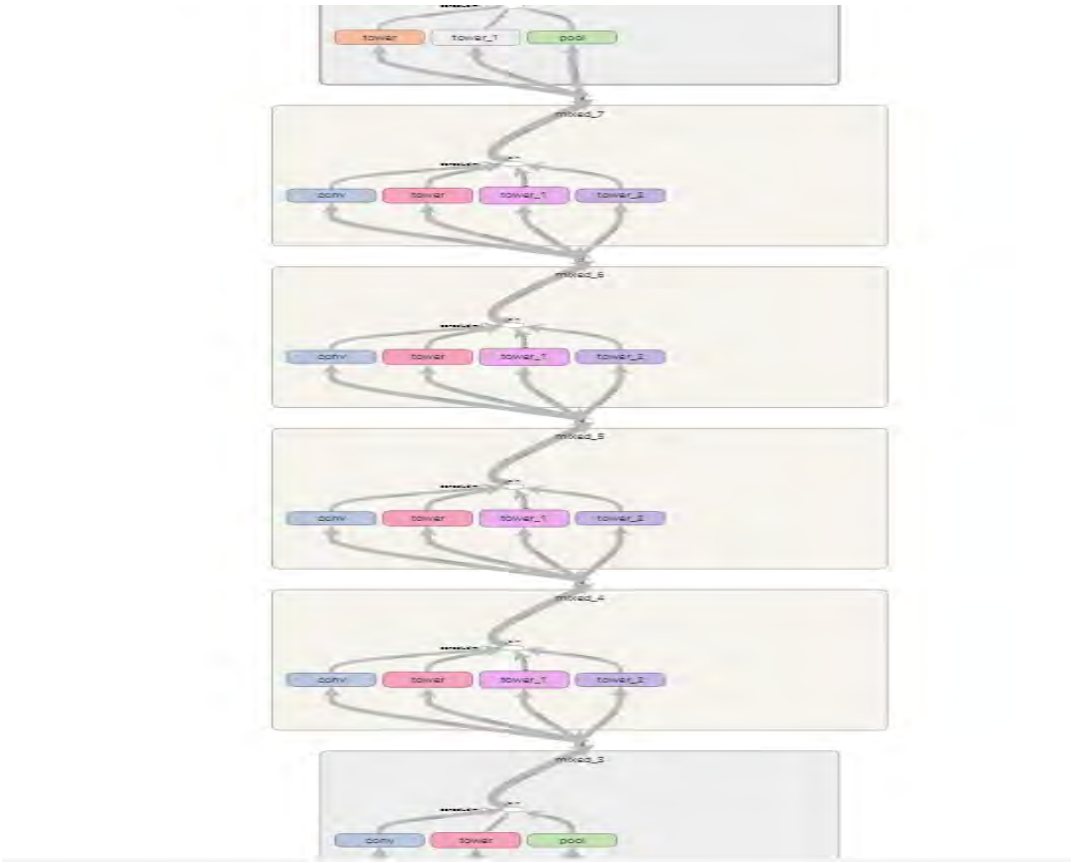
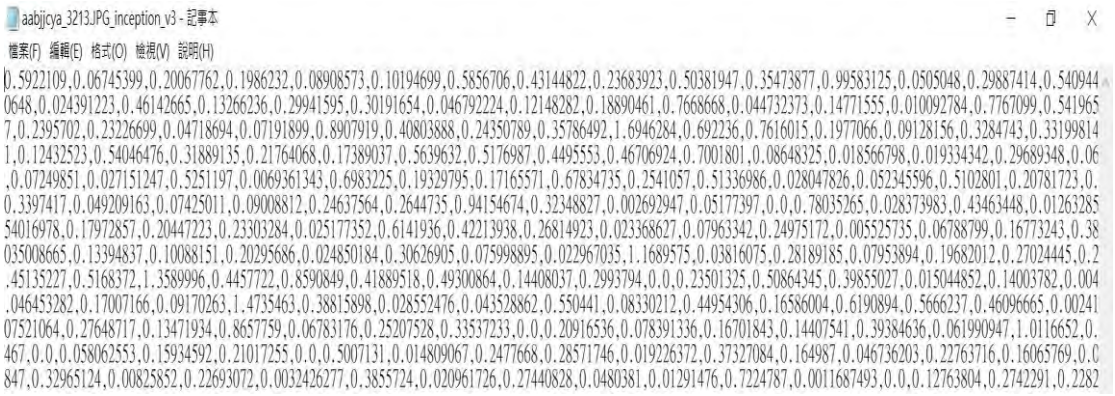


圖 12 下載的 InceptionV3 模型之部分計算圖

2. 將照片作為輸入利用 sess.run()經過模型計算後產生輸出 ---bottleneck file(如圖 13)。內容為圖片經 InceptionV3 的第一個 convolutional layer 到 dropout 前的 avg_pool layer 的輸出結果。


aabjjcya_3213.JPG_inception_v3 - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

0.5922109,0.06745399,0.20067762,0.1986232,0.08908573,0.10194699,0.5856706,0.43144822,0.23683923,0.50381947,0.35473877,0.99583125,0.0505048,0.29887414,0.540944
0648,0.024391223,0.46142665,0.13266236,0.29941595,0.30191654,0.046792224,0.12148282,0.18890461,0.7668668,0.044732373,0.14771555,0.010092784,0.7767099,0.541965
7,0.2395702,0.23226699,0.04718694,0.07191899,0.8907919,0.40803888,0.24350789,0.35786492,1.6946284,0.692236,0.7616015,0.1977066,0.09128156,0.3284743,0.33199814
1,0.12432523,0.54046476,0.31889135,0.21764068,0.17389037,0.5639632,0.5176987,0.4495553,0.46706924,0.7001801,0.08648325,0.018566798,0.019334342,0.29689348,0.06
,0.07249851,0.027151247,0.5251197,0.0069361343,0.6983225,0.19329795,0.17165571,0.67834735,0.2541057,0.51336986,0.028047826,0.052345596,0.5102801,0.20781723,0.
0.3397417,0.049209163,0.07425011,0.09008812,0.24637564,0.2644735,0.94154674,0.32348827,0.002692947,0.05177397,0.0,0.78035265,0.028373983,0.43463448,0.01263285
54016978,0.17972857,0.20447223,0.23303284,0.025177352,0.6141936,0.42213938,0.26814923,0.023368627,0.07963342,0.24975172,0.005525735,0.06788799,0.16773243,0.38
035008665,0.13394837,0.10088151,0.20295686,0.024850184,0.30626905,0.075998895,0.022967035,1.1689575,0.03816075,0.28189185,0.07953894,0.19682012,0.27024445,0.2
.45135227,0.5168372,1.3589996,0.4457722,0.8590849,0.41889518,0.49300864,0.14408037,0.2993794,0.0,0.23501325,0.50864345,0.39855027,0.015044852,0.14003782,0.004
.046453282,0.17007166,0.09170263,1.4735463,0.38815898,0.028552476,0.043528862,0.550441,0.08330212,0.44954306,0.16586004,0.6190894,0.5666237,0.46096665,0.00241
07521064,0.27648717,0.13471934,0.8657759,0.06783176,0.25207528,0.33537233,0.0,0.20916536,0.078391336,0.16701843,0.14407541,0.39384636,0.061990947,1.0116652,0.
467,0.0,0.058062553,0.15934592,0.21017255,0.0,0.5007131,0.014809067,0.2477668,0.28571746,0.019226372,0.37327084,0.164987,0.046736203,0.22763716,0.16065769,0.0
847,0.32965124,0.00825852,0.22693072,0.0032426277,0.3855724,0.020961726,0.27440828,0.0480381,0.01291476,0.7224787,0.0011687493,0.0,0.12763804,0.2742291,0.2282

圖 13 bottleneck file 部分內容

3. 將 bottleneck file 儲存起來，因為計算出 bottleneck 數值為訓練過程中最耗時的部分之一。

4. 將 bottleneck file 中之數值作為最後全連階層的訓練輸入。

5. 正向傳播後進行反向傳播算出誤差，再使用梯度下降演算法更新參數，

以上演算法非本研究主軸故不加以贅述。

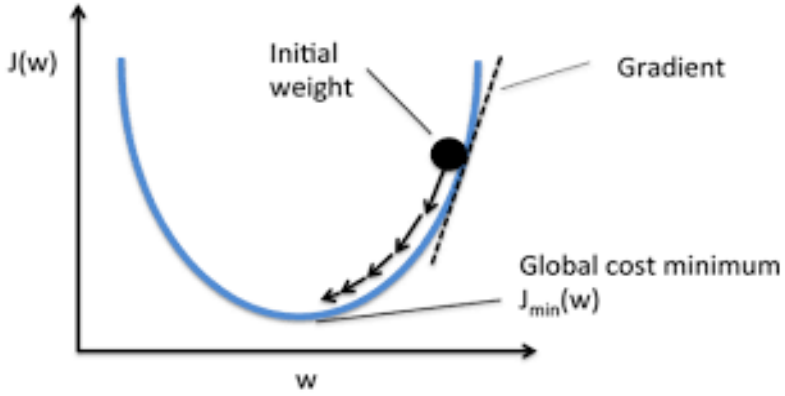


圖 14 梯度下降演算法示意圖
圖片來源：Batch gradient descent vs
Stochastic gradient descent

6. 儲存訓練後的計算圖(及權重)

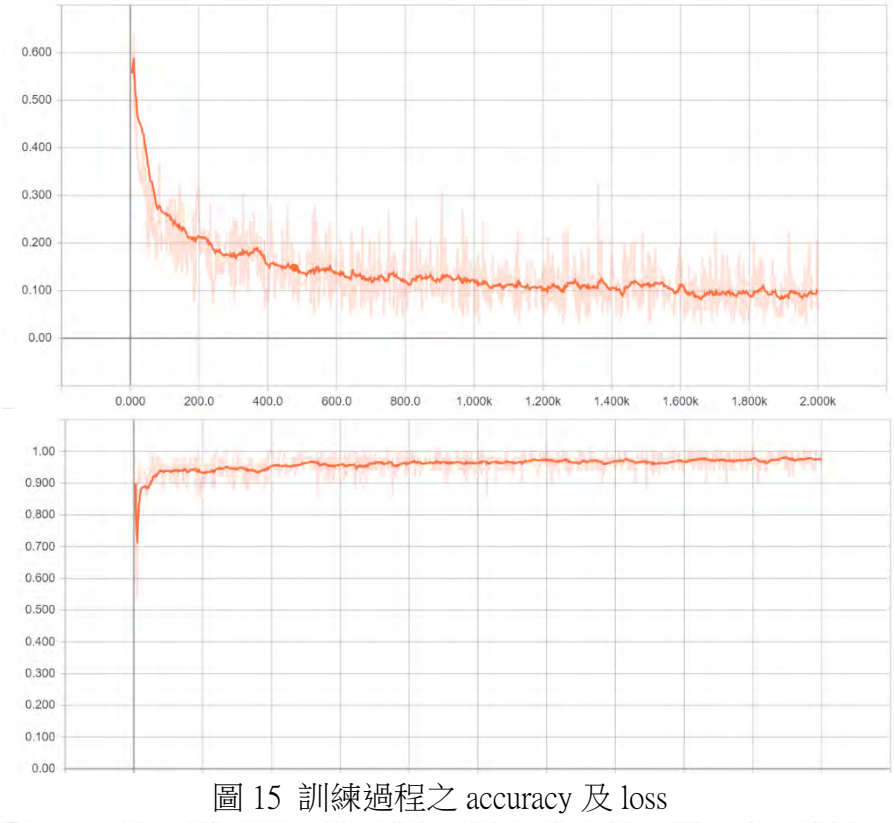


圖 15 訓練過程之 accuracy 及 loss

7. 輸入圖片，根據輸出來判斷結果。

五、手機 APP

本次編撰手機程式採用的是 Android Studio，使用的程式語言為 Java。

這部分的手機程式，讓使用者能即時監控。這支程式主要由五大部分組成，如表 2

表 2 手機程式架構介紹表

類別(Class)	功能	關係
JustService	收發訊息及傳送通知。	-
MainActivity	為首頁，有多個按鈕，能顯示門前訪客，傳送開關門指令及察看歷史紀錄。	按下"CHECK ACTIVITY LOG"後，前往 ActivityLog 察看歷史紀錄。
ActivityLog	儲存來訪過的人物，作為紀錄，以及一按鈕，按下顯示樹莓派拍下的照片。	按下"MORE"後，前往 Popout 察看樹派拍下的照片。
Popout	將 encode 過的圖片字串 decode 回原始圖檔。	-
Notifications	宣告通知之 Channel ID，為建立通知的前置作業。	-

(一) JustService

JustService 負責在背景執行所有收發訊息的作業，以及傳送通知給使用者。最一開始，此類別(Class)會繼承(Extends)服務(Service)類別，使其能在背景執行作業。同樣是藉由 Websocket 的方式，使用預設好 Server 的 IP 和連接埠(Port)。透過 onMessage 函式取得收到的訊息，並於此函式中設計幾個條件判斷。如果收到的訊息為"None"，即門前無人，則不做任何事；若結果不為”None”，則於此訊息字串末端加上當前時間，並且將其存入一陣列之中及傳送通知，亦即使用者在收到通知前往察看時，可以得知門前的人是誰，和出現時間點。

```
public void onMessage(String s) {
    final String message = s;
    notifmessage = message;
        if(message.equals("None"))//若 Server 傳的訊息為 None
    {
        if(!situation.equals("None"))
            //更新在 APP 首頁的”目前狀態”欄位
            situation="Nobody is at the door currently";
    }
    //若 Server 傳的訊息不為 None
    else {
        //若 Server 傳的訊息長度介於 3~60 之間 //self-explanatory
        if (s.length() < 60 && s.length() > 3) {
            situation=message;
           //將門前人物的身分加上時間後處存於陣列中
            ActivityLog.sb_comes.add(message+" "+currentTime);
            cal++;
            //傳送通知
            sendNotifications();
        }
        //若 Server 傳的訊息長度不介於 3~60 之間，則為 encode 過後的圖片
        else {
            Popout.sb_img.add(message);
        }
    }
}
```

圖 16 JustService 部分程式流程圖

(二) MainActivity

MainActivity 管理的頁面為使用者登入此 APP 的首頁。此介面由四個按鈕構成：” Refresh” ，” Keep Locked"，” Unlock”和” Check Activity Log”

1. Refresh：按下後會在隔壁顯示門前最新的活動。
2. Keep Locked 及 Unlock:可以遠端控制門的開啟與關閉。
3. Check Activity Log:打開 ActivityLog 介面，查詢歷史紀錄。

圖 17 APP 首頁(紅框內為按鈕)

(三) ActivityLog

ActivityLog 內存有門前所有出現的訪客，如歷史紀錄般。左邊顯示的是在 JustService 中儲存於陣列中的訊息，也就是門前出現過的人，以及出現的時間；右邊的按鈕則是按下後，會進到 Popout 介面，顯示樹莓派拍下圖片，以供使用者再次確認。

圖 18 ActivityLog 視窗畫面

```
for (int i = till; i < cal ; ++i) {
        LayoutInflater layoutInflater =
                (LayoutInflater)
getBaseContext().getSystemService(Context.LAYOUT_INFLATER_SERVI
CE);

        final View addView = layoutInflater.inflate(R.layout.row, null);
        final TextView textOut = (TextView)addView.findViewById(R.id.textout);
        textOut.setText(sb_comes.get(i));
        final Button buttonView = (Button)addView.findViewById(R.id.more);
        till = i;
        buttonView.setTag(i);
        //設定按下 More 之後的動作
        buttonView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //前往 Popout 彈出視窗  // Pops up Popout screen with image
                Intent intent = new Intent(ActivityLog.this, Popout.class);
                //傳送每個按鈕對應的圖片 index 傳至 Popout activity
                intent.putExtra("BMI_EXTRA", (int)buttonView.getTag());
                startActivity(intent);
            }});
        //將一個 row 加入顯示介面
        container.addView(addView, 0);
}
```

圖 19 ActivityLog 部分程式

(四) Popout

Popout 負責將被 encode 的圖片字串，decode 回一張完整的圖片，在按鈕被按下後，彈出含有照片的此視窗，使用者即可判斷是否要執行門的開關。

```
//將 Server 傳來 encode 後的圖片 decode 回原始圖檔 //decodes encoded image sent from Server，sb_img 為一儲存編碼後圖片之陣列
byte[] decodedString = Base64.decode(sb_img.get(num), Base64.DEFAULT);
Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString, 0, decodedString.length);
img.setImageBitmap(decodedByte);
```

圖 20 Popout 部分程式

(五) Notifications

此類別係因 Android8.0(API26)以上，在使用 NotificationCompat.Builder 建立通知時需要提供參數 Channel ID。因此，建立另一個類別 Notifications.java，宣告 Channel ID，而主要建立通知的函式則是在 JustService 類別下宣告。以下為 Notifications.java 的部分程式碼。

```
private void createNotificationChannels(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
        NotificationChannel channel = new NotificationChannel(
                Channel_1_ID,
                "Channel 1",
                NotificationManager.IMPORTANCE_HIGH
        );
        channel.setDescription("This is Channel 1");
        NotificationManager manager = getSystemService(NotificationManager.class);
        manager.createNotificationChannel(channel);
    }
}
```

圖 21 Notification 程式

以下為在 JustService 下宣告的傳送通知的函式。

```
public void sendNotifications(){
    Notification notification = new NotificationCompat.Builder(this,
Notifications.Channel_1_ID)
                .setSmallIcon(R.drawable.ic_face_black_24dp)
                .setContentTitle("MESSAGE FROM iDOOR")
                //在通知中告知使用者門前的人為誰
                .setContentText(""+ notifmessage + " is at the door!")
                .setCategory(NotificationCompat.CATEGORY_MESSAGE)
                .build();
    notificationManager.notify(1, notification);
}
```

圖 22 sendNotification 函式

六、使用者操作

這部分負責處理使用者收到門前的活動後，可以再決定是否該保持門的關閉。

1. 手機端：

   由 MainActivity 的兩個按鈕組成：Keep Locked 和 Unlock。在按下任一按鈕後，會分別傳出 0 或 1，以供樹莓派執行。

2. 樹莓派端：

   負責此部分的程式為前面所提的 Final.py。首先會先連接上 Server，在收到 0 或 1 後，會分別對伺服馬達下不同的指令，0 為維持在 0 度；1 則讓伺服馬達轉至 90 度，模仿門打開的效果。

伍、 研究結果

以下為本次研究最終產出的裝置。圖 25 為門眼的特寫圖，殼內為樹莓派、紅外線感測器、攝影機和麵包版及線路。圖 23 為伺服馬達，連接門眼內的樹莓派，之後會裝上模

圖 23 伺服馬達

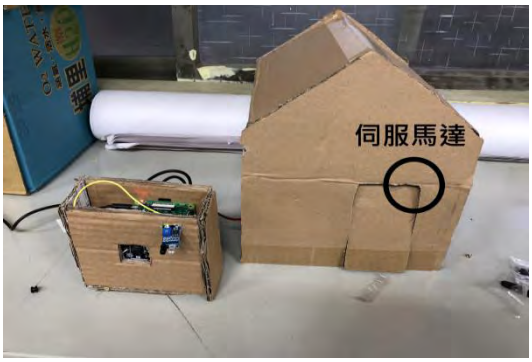


圖 24 整體裝置(門眼與模型屋)

圖 25 門眼特寫圖

擬的小門，使用者可以由手機傳來的訊息決定是否要轉動它。圖 24 為整個裝置在配置完成後，完整的樣貌。

本次研究訓練的資料集包含 3~5 人的影像，希望可以判斷來訪訪客的身分。之後也會加上”Anonymous”的資料集，以便判斷未知訪客的到來。

以下將實際模擬此系統在現實生活中會遇到的幾種狀況：

- 狀況一：門前無人的時候。
- 狀況二：門前有人，則判斷其身分。

一、門前無人時，亦即正常狀況

如圖 26，可以看到樹莓派在下方 Shell 中顯示”No intruders”表示沒有感測到人

圖 28 中右方為電腦終端機執行 Server 程式時的樣貌，而圖 27 則為 toServer 的執行畫面，其中，終端機不斷顯示”None”代表 Server 收到 toServer 傳的訊息，說明門前無人。

圖 26 樹莓派顯示無感測到人

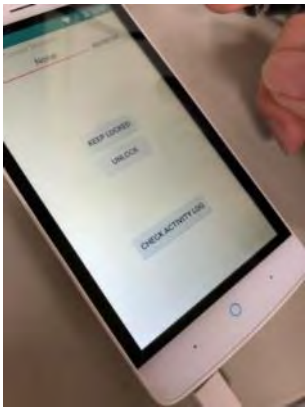


圖 27 手機介面

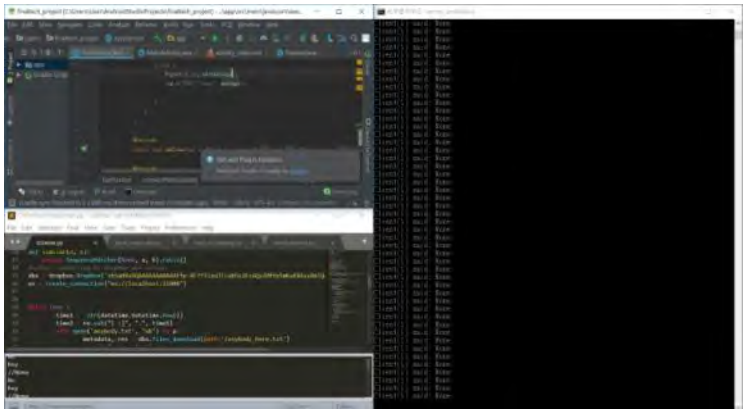


圖 28 電腦執行畫面

二、門前有人則進行判斷。

如圖 29，可以看到樹莓派在下方 Shell 中顯示”Intruder detected”表示有感測到人本次門前人物為 JXXXX。在樹莓派感應到有物體出現之後，將拍下照片上傳至 Dropbox 供 toServer 辨識。辨識結果為 JXXXX，於是結果和照片將傳至 Server 再傳至手機。此處圖 30 為按下 REFRESH 鍵後，門前人物將顯示於上端 Current Situation 下。圖 31 為按下 CHECK ACTIVITY LOG 後所彈出的歷史紀錄視窗，由圖中可看到目前有一人來訪過，人物為誰和來訪時間點。圖 32 則為按下 MORE 後，秀出樹莓派拍下的圖片 。

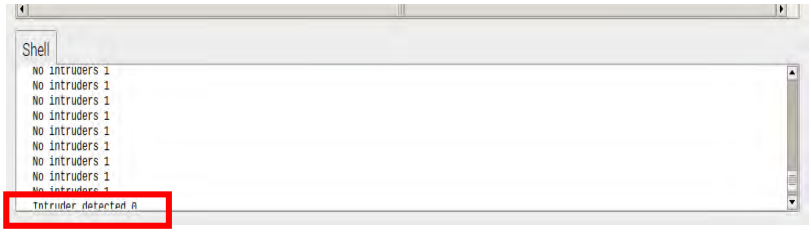


圖 29 樹莓派顯示感測到人

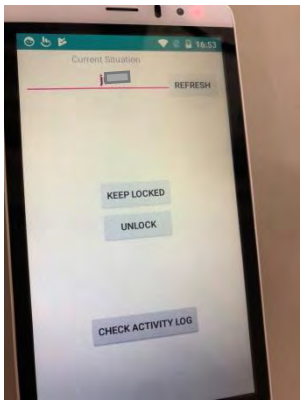


圖 30 手機程式首頁

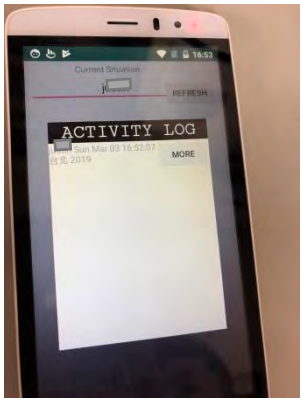


圖 31 Activity Log 介面

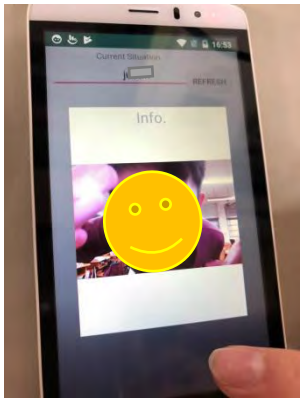


圖 32 手機顯示門前影像

最後，以下為驗證使用者在按下 KEEP LOCKED/UNLOCK 鍵後的反應是否達到預期。如上述所說，按下前者會傳送至 Server，而 Server 再將訊息傳給所有使用者。樹莓派

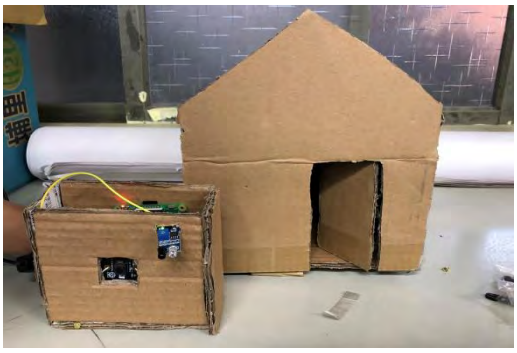


圖 37 門解鎖並打開

在收到訊息後將作出與其相對的反應。左圖為按下 KEEP LOCKED 鍵後，伺服馬達將維持於 0 度；又圖則為按下 UNLOCK 鍵後，伺服馬達將轉 90 度，進而使門打開。

四、辨識結果測試：

本研究試著從辨識兩人開始試著往上訓練，下表為針對不同人數作辨識的表現。目前正努力增加資料集內人數，並嘗試多種其他不同模型，ex.VGG。

表 3 人臉辨識內容及結果

	2 人	3 人	4 人	5 人
辨識類別	Justin/eagle	Justin/eagle/brian	Justin/eagle/brian /bear	Justin/eagle/brian /bear/chosen
訓練照片數	每人各 50 張			
測試照片數	每人各 10 張			
測試 acc	18/20=90%	27/30=90%	37/40=92.5%	40/50=80%

判斷錯誤的照片某些是因為在極暗處拍攝的或模糊的外，大致上可以透過增加資料的多變性來解決，目前也正在著手處理當中。

陸、 討論

一、關於感應器的選擇…

一開始使用的是 PIR 感應器(Passive Infrared)，PIR 感測器是利用熱量的變化，因此，當有人站在門前不動時就可能有感測不到的情形。所以後來的實作中，才換成使用主動式紅外線感應器。主動式紅外線感應，會發出紅外線，當有人在門前時，感應器就會收到反射回來的信號而產生高輸出，因而感應到有人位於門前。

圖 38 PIR 感應器

二、關於攝影機的選擇…

Pi Camera 是由 Raspberry Pi 提供的攝影機。使用 Pi Camera 而非一般 USB Webcam 是因為 Pi Camera 直接連接到的是樹莓派的 GPU，能夠提供比較順暢的畫面且對 CPU 產生相對較低的負荷量，又不失提供高畫質的照片。另外，此型號的攝影機有提供夜視功能，在外掛特定影體設備後，能夠在缺乏光線時也拍出門前的景象。

三、影像辨識研究過程之討論

最初使用的深度學習工具為 Keras

Keras 是基於 Python 高階深度學習的程式庫，為 Tensorflow 的高階 API。

之所以選擇 Keras 作為起手，是因為 Keras 用起來十分平易近人，比如許多程式用 Tensorflow 需要很多行撰寫，但在 Keras 中變得只需一行即可達成。

1. Autoencoder

最初想使用的機器學習方法是 Autoencoder，Autoencoder 是將圖片進行編碼再解碼，所產生的 Output 與原圖片比較並進行反向傳播(Back Propagation)

最一開始選擇 Autoencoder 是因為當時最一開始的目標為辨識「是否」為某人，而 Autoencoder 正好可以產生這樣一對一的輸出，一般的辨識則大部分都需要兩個類別以上來辨別。

首先使用了 MNIST 手寫資料集做測試，效果不錯，如圖 40；但之後在用人臉的照片來測試時，效果就沒有那麼好了，如圖 41。

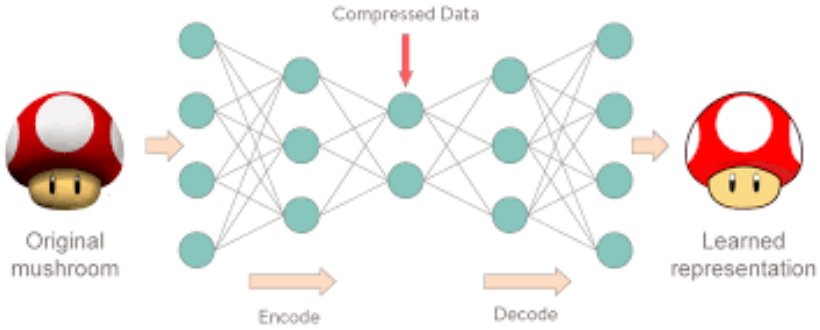


圖 39 Autoencoder 示意圖

(圖片來源：https://towardsdatascience.com/deep-autoencoders-using-tensorflow-c68f075fd1a3)

圖 40 Autoencoder 在 MNIST 資料集上的表現

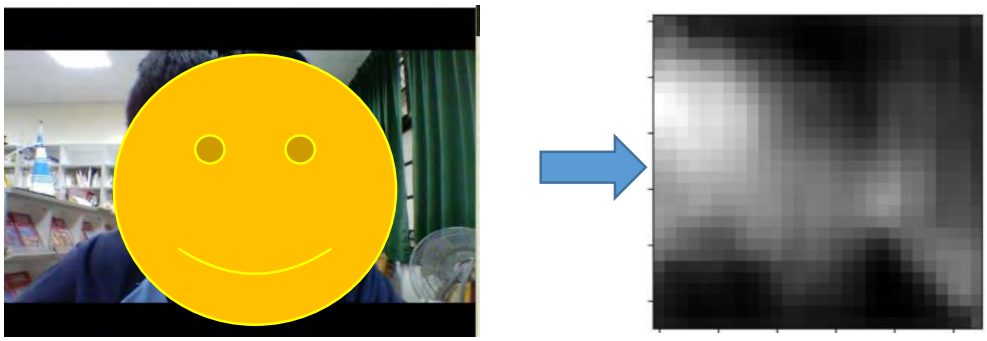


圖 41 Autoencoder 在人臉照片上的表現

2. CNN

之後，本研究試著使用 Keras 之函式庫，從頭訓練一個順序(Sequential)的模型來判別為 Justin 或 Eagle。首先，將"Justin"及"Eagle"的圖片讀取為灰階後存入 feature 陣列，然後將 labels("Justin"為 0 或"Eagle"為 1)存入 label 陣列，使每張照片對應到一個正確解答(label)，且轉為指定大小。

接著，嘗試的建立了一個三層捲積層的 CNN Sequential 模型，正向傳播過程中，以"ReLU"作為起動函數以實現非線性轉換，最後一層則使用"Sigmoid"作為啟動函數，如此一來，便只要判斷輸出的結果接近 1 或 0 即可判斷照

```
model = Sequential([
    Conv2D(64, (3, 3), input_shape=features.shape[1:], padding='same',
           activation='relu'),

    Conv2D(256, (3, 3), activation='relu', padding='same'),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Dropout(0.25),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

圖 42 使用 Keras 建立 Sequential 模型

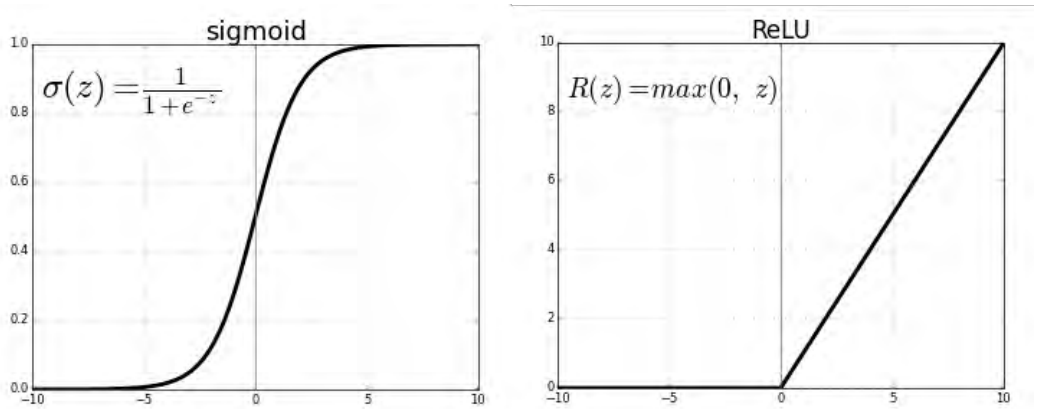


圖 43 ReLU 與 Sigmoid 非線性轉換示意圖

片為"Justin"或"Eagle"。

起初，訓練的結果不錯，但測試時結果不夠準確，於是逐步增加層數，但是，隨層數的增加，參數的數量就越多，電腦跑起來就十分緩慢，且資料的需求量就越大，訓練出來的成果就不太理想，這邊就不放上準確率測試。

```
6500/7503 [=======================>.....] - ETA: 41s - loss: 7.2333 - acc: 0.5512
6600/7503 [=======================>.....] - ETA: 37s - loss: 7.2409 - acc: 0.5508
6700/7503 [=======================>.....] - ETA: 33s - loss: 7.2363 - acc: 0.5510
6800/7503 [========================>....] - ETA: 29s - loss: 7.2508 - acc: 0.5501
6900/7503 [========================>....] - ETA: 25s - loss: 7.2461 - acc: 0.5504
7000/7503 [========================>....] - ETA: 21s - loss: 7.2554 - acc: 0.5499
7100/7503 [========================>....] - ETA: 17s - loss: 7.2600 - acc: 0.5496
7200/7503 [========================>....] - ETA: 13s - loss: 7.2531 - acc: 0.5500
7300/7503 [========================>....] - ETA: 9s - loss: 7.2598 - acc: 0.5496
7400/7503 [========================>....] - ETA: 4s - loss: 7.2597 - acc: 0.5496
```

圖 44 訓練部分過程

四、額外新增功能

1.相較於上個版本，本次於使用者手機 APP 上多了「即時畫面」的功能，如圖 45 所示，按下此”REAL TIME FOOTAGE”將顯示一彈出視窗如圖 46，畫面為樹莓派及時上傳到特定網址的即時影片，用以作為使用者確認門前狀況的功能。

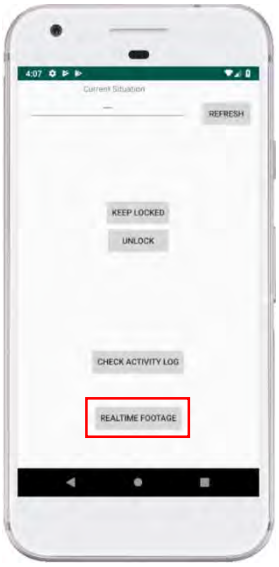


圖 45 新版使用者介面

圖 46 門前即時影像

2.未來，我們將會於手機 APP 上新增一功能，它能拍下新的訪客影片，將它納入此系統的影像辨識資料集內。猶如 iphone 的 Face ID 功能一樣，透過使用者拍下欲新增的訪客照片，即能使之也能輕鬆進入門內。

柒、 結論

本次的智慧門禁系統與過往的傳統門禁系統有別。由樹莓派拍下訪客的照片，交給電腦進行影像辨識，將結果傳到使用者手中，再由他/她決定是否為其開門。 這樣的系統省去了很多過去繁瑣的過程，影像辨識加快了傳統需要透過門眼察看，或主人不在家時需要透過通話確認的步驟。

此外，有鑑於現在手機的普及，本研究結合手機應用程式，能夠即時通知使用者訪客的到來，清楚顯示在 APP 中，讓使用者能一目了然。最後，門禁系統最注重的就是安全性，本次研究在模擬的時候，門不會無故開啟，而若使用者擔心影像辨識結果有誤，還能察看樹莓派拍下的照片做為確認。

此系統也能用於其他如保險箱、置物櫃等其他有「門」之設備，未來我們將會開始研究 Deepid、fisherface 等人臉辨識演算法，增加系統的準確率，甚至讓特定人物能經過辨識直接進門(即「臉就是鑰匙」)，同時增加系統可識別的人數，並探索增加系統靈活性及穩定性之方法，希望能夠應用在校習得的觀念，替大眾製作出能讓生活更便捷的作品。

捌、 參考文獻

(一) Bill Butterfield Android Studio Tutorials

https://www.youtube.com/channel/UCTfCl-a8_6aKT_Mdd4HkaUw

(二) Alexander Baran-Harper Raspberry Pi Tutorials

https://www.youtube.com/channel/UC_aQTJgfrnCb8coPbZ5cgJw/playlists

(三) Dropbox for Python Documentation

https://dropbox-sdk-python.readthedocs.io/en/latest/

(四) upload a file to Dropbox from a Raspberry Pi via Python

https://www.youtube.com/watch?v=40D6fP9GXU4

(五) Raspberry Pi 嵌入式系統入門與應用實作 作者：張元翔 出版社：碁峯

(六) 精通 Python 作者：Bill Lubanovic 譯者：賴屹民 出版社：碁峯

(七) Tensorflow 官網:https://www.tensorflow.org/

(八) 看的懂的人工智慧 作者:鄭澤宇、梁博文、故斯與 出版社:深石

(九) https://github.com/websocket-client/websocket-client

(十) https://github.com/Pithikos/python-websocket-server

(十一) https://github.com/varvet/mobile-websocket-example

(十二) Keras documentation：https://keras.io

【評語】052502

此作品是個物聯網應用，利用 Pi 連接紅外線感測器來感測是否有人站在門口，連接攝影機拍攝人臉影像，把多人的人臉影像先進行訓練，再把拍攝到的人臉影像用訓練出來的模型進行辨識，若同意開門則送命令給 Pi 要它啟動步進馬達旋轉門把開門。

此研究主題缺乏創新性，作品以實作為主，對於科學探究的實驗評估不夠充分。如同計畫書中作者所說，人臉影像辨識這部分只是拿現成的軟體來用，並沒有提出能提升辨認成功率的方法。此外就算目前只以少數 (5 人以內)人數進行實驗，辨識率仍然不高，距離實用還有很遠的距離。

建議未來多加強在多人情況下的人臉影像辨識這部分的正確率，可讓作品更有實用價值。

作品海報

壹、摘要

本研究為方便及簡化現有門禁系統繁複的流程，透過物聯網、深度學習和手機應用程式的組合，來實現這樣的效果。在門前用樹莓派拍下訪客的照片，之後再將之傳至電腦，透過移轉學習訓練完成的神經網路模型，對訪客照片進行辨識，輸出初步的判定結果，並將其傳給主人的手機，經由Android Studio設計的app顯示，最後由主人透過遠端操控，決定最終的開門判斷。

貳、研究動機

傳統的門禁系統的流程十分繁瑣，訪客通常得要先按門鈴，然後在門前等候主人來應門，等到主人匆忙趕到門後，還得透過門眼察看訪客的身分後才決定是否進行開門的動作。本研究希望能簡化開門的流程，製作出一套方便，又不失安全性的現代門禁系統。

參、研究目的

本次研究之目的，希望結合影像辨識、IoT及深度學習技術，藉日常所學設計一套智慧型的門禁系統。當有訪客來訪時，本系統能判別訪客的身分，並將照片及辨識結果儲存至主人的手機，再由主人決定是否開門。當主人決定替訪客開門時，可透過設計過的手機app，進行遠端操控完成開門程序。

肆、研究設備

一、硬體設備：筆電、Raspberry Pi 3樹莓派、IR sensor module紅外線感應模組、Raspberry Pi Camera NoIR 攝影機伺服馬達 MG946R、 TWM Amazing X3 手機
二、軟體設備：Android Studio、Sublime Text、VNC Viewer、Python 3.5 (64-bit)
三、深度學習之函式庫：tensorflow1.9.0, keras2.2.4

伍、研究過程

一、研究過程
(一) 資料蒐集:蒐集機器學習所需的資料集
(二) 程式撰寫:撰寫樹莓派、資料傳遞、手機App(Android Studio)的程式及測試影像辨識的方式
(三) 結果檢視:執行拍照、辨識到上傳至手機的過程

二、系統架構

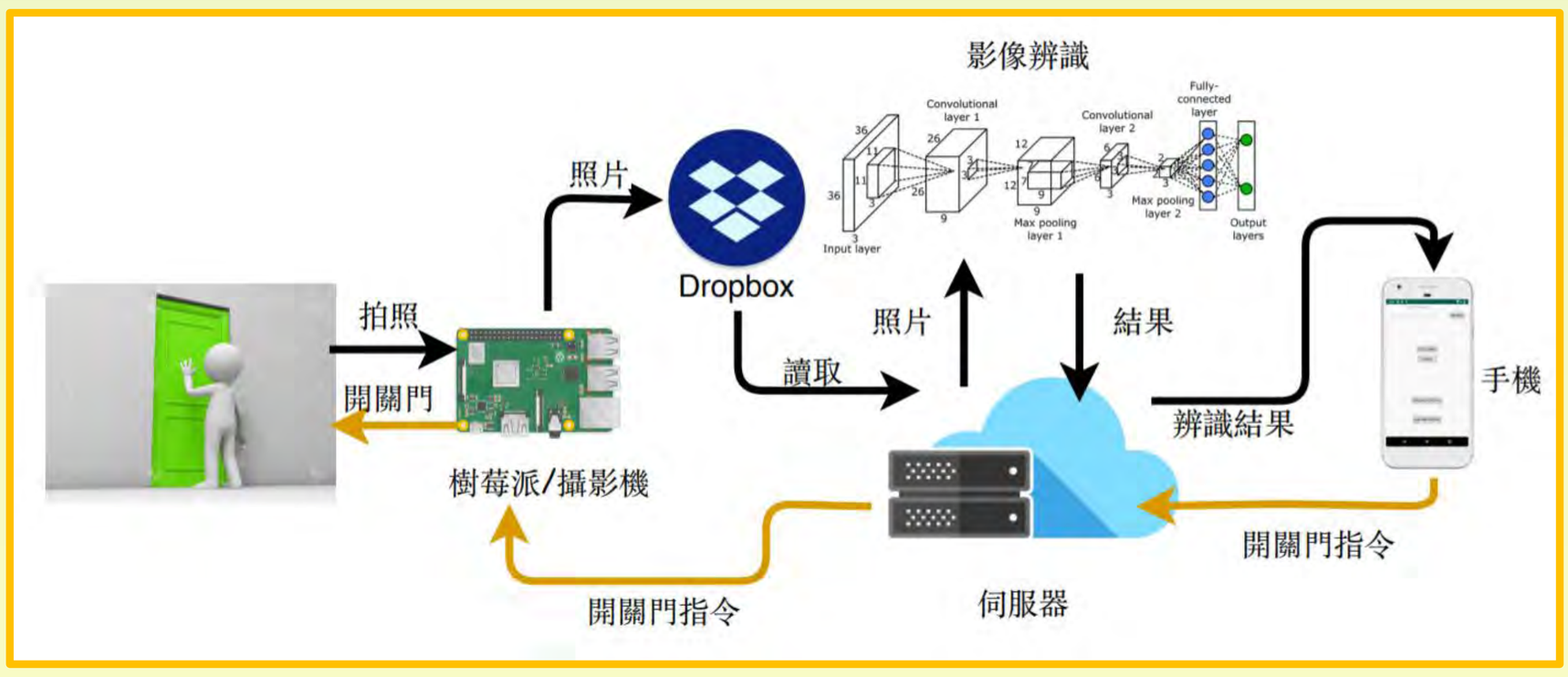


圖1 系統架構

(一)若感應到物體，樹莓派(門眼端)擷取門前訪客之影像
(二)上傳影像並由電腦下載
(三)電腦執行影像辨識(本次以辨識Justin/Eagle作為試驗)
(四)將辨識的結果及影像傳至手機供使用者作判斷
(五)使用者決定要開門或關門

三、影像辨識之訓練資料

- 光影變換
- 不同角度
- 每人各25張
  （justin & eagle）

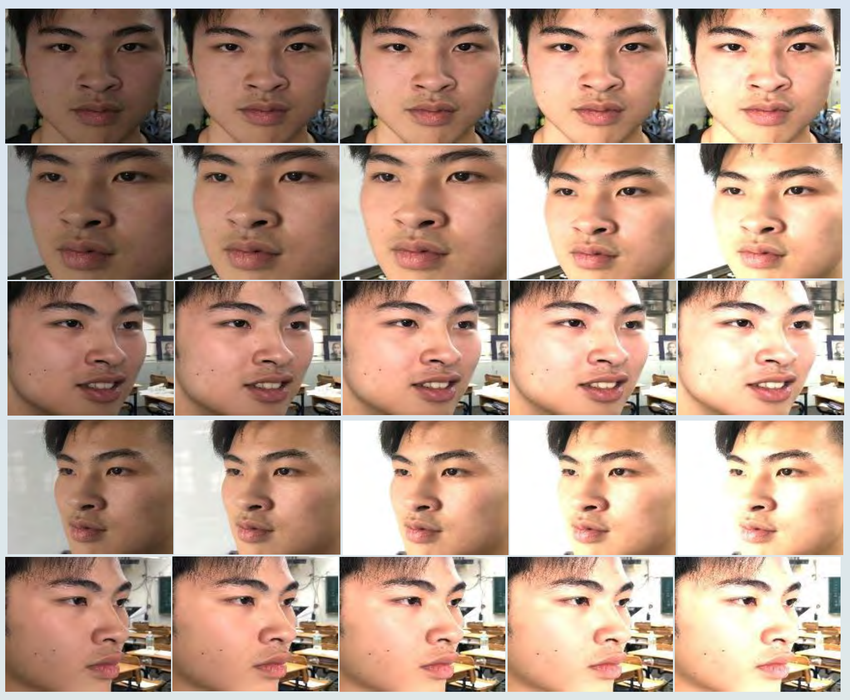
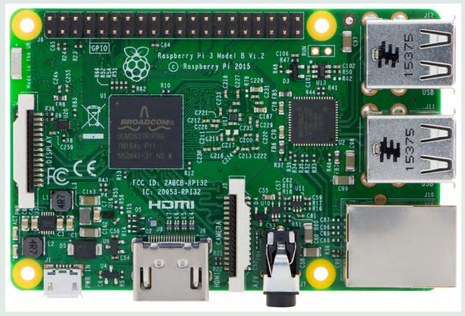


圖2 eagle訓練資料集

四、程式介紹



程式名稱	功能	語言	連結
Initial.py	1. 感應訪客 拍下影像 2. 將影像回傳至電腦	Python	提供影像辨識的輸入
Final.py (Client)	1.接收電腦指令 2.執行開關門	Python	執行使用者下達的開關門指令

➢ 表1 樹莓派程式

程式名稱	功能	語言	連結
toServer.py (Client)	1.辨識訪客影像 2.傳送辨識結果至Server	Python	初步輸出辨識結果
Server.py (Server)	1.接收所有Client的訊息 2.修飾訊息後發送給其他Client	Python	所有Client訊息的轉運站

➢ 表2 樹莓派程式

編輯環境	功能	語言	連結
Android Studio	顯示訪客資訊和提供使用者遠端操作的介面	Java	人與系統互動的介面

➢ 表3 手機程式

陸、研究結果

➢樹莓派

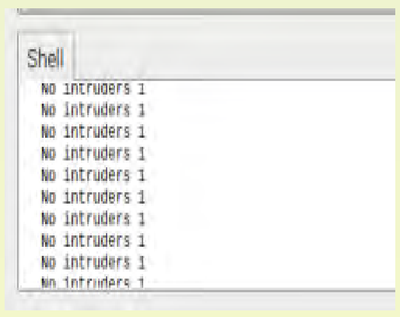


圖3 門前無人

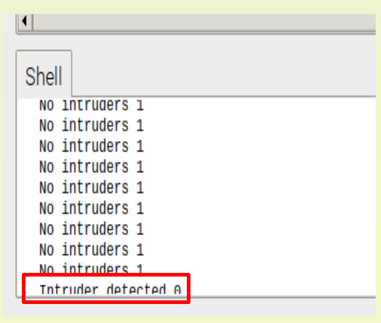


圖4 門前無人

➢電腦

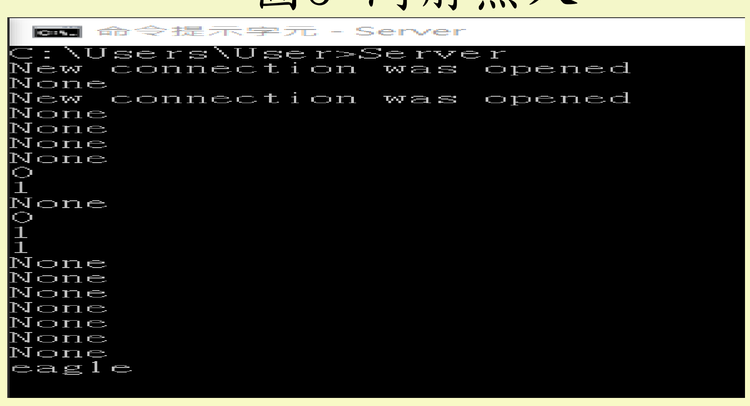


圖5 Server執行畫面

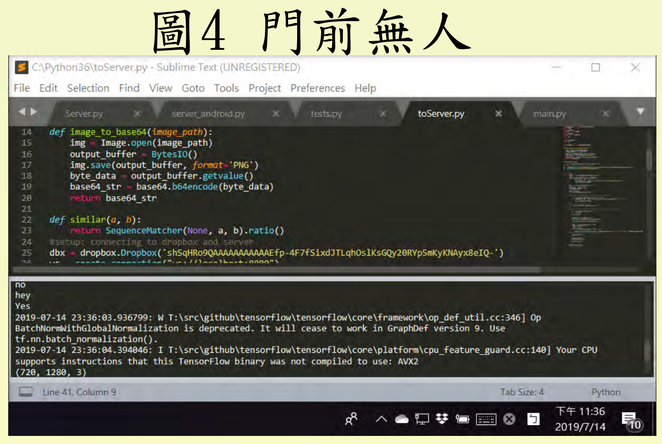
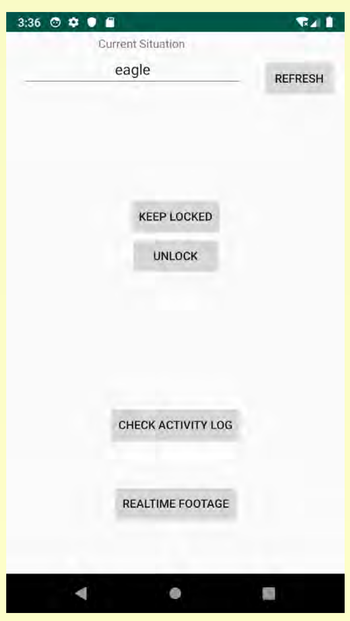
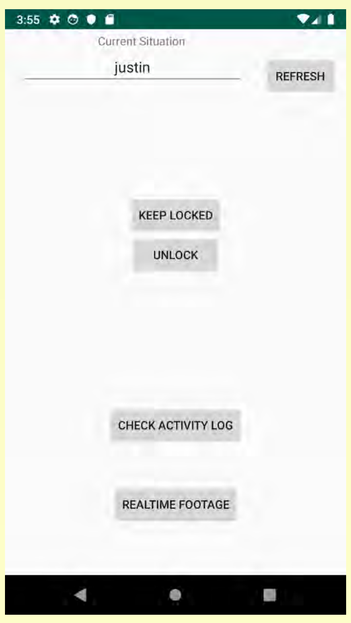


圖6 toServer執行畫面

➢手機





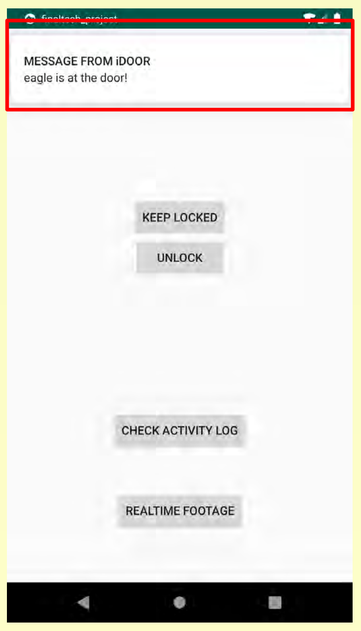
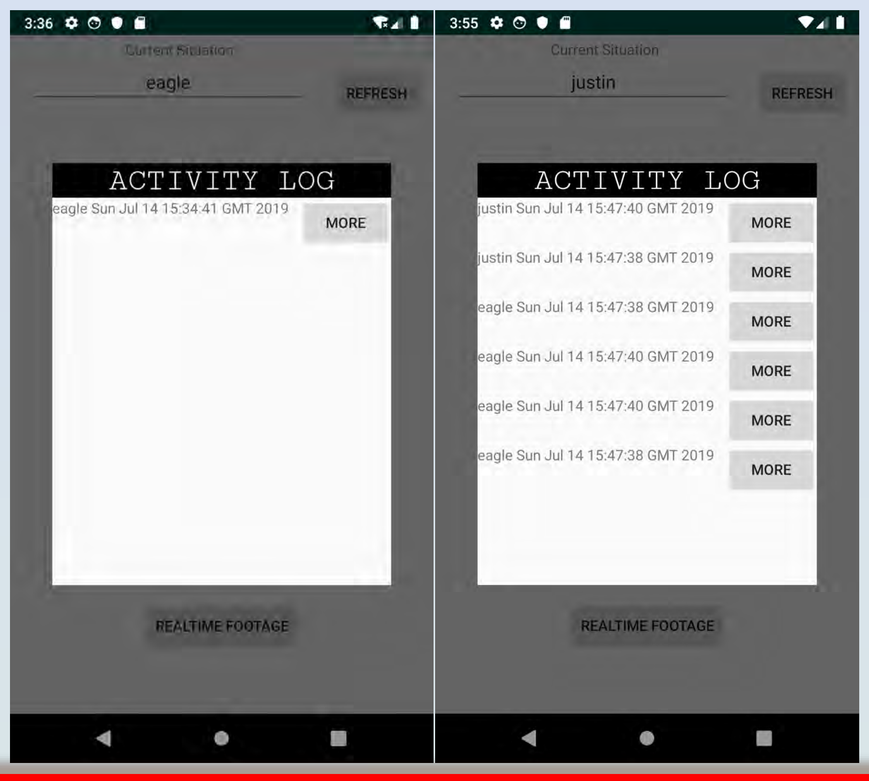
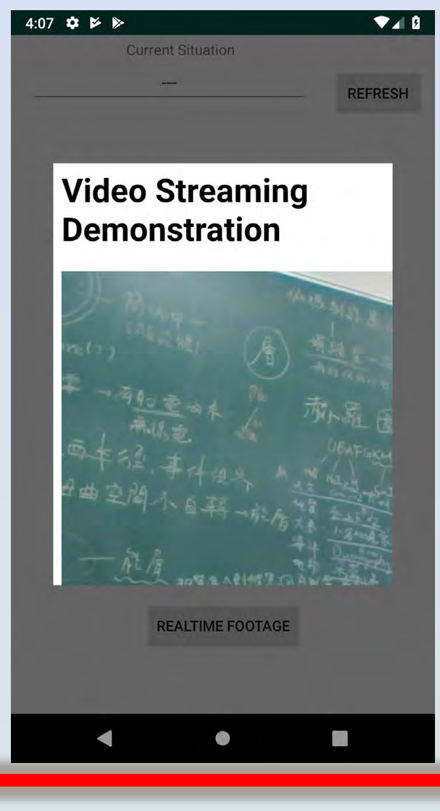


圖7 訪客為eagle　圖8 訪客為justin　圖9 顯示通知

➢ 圖10
訪客歷史紀錄

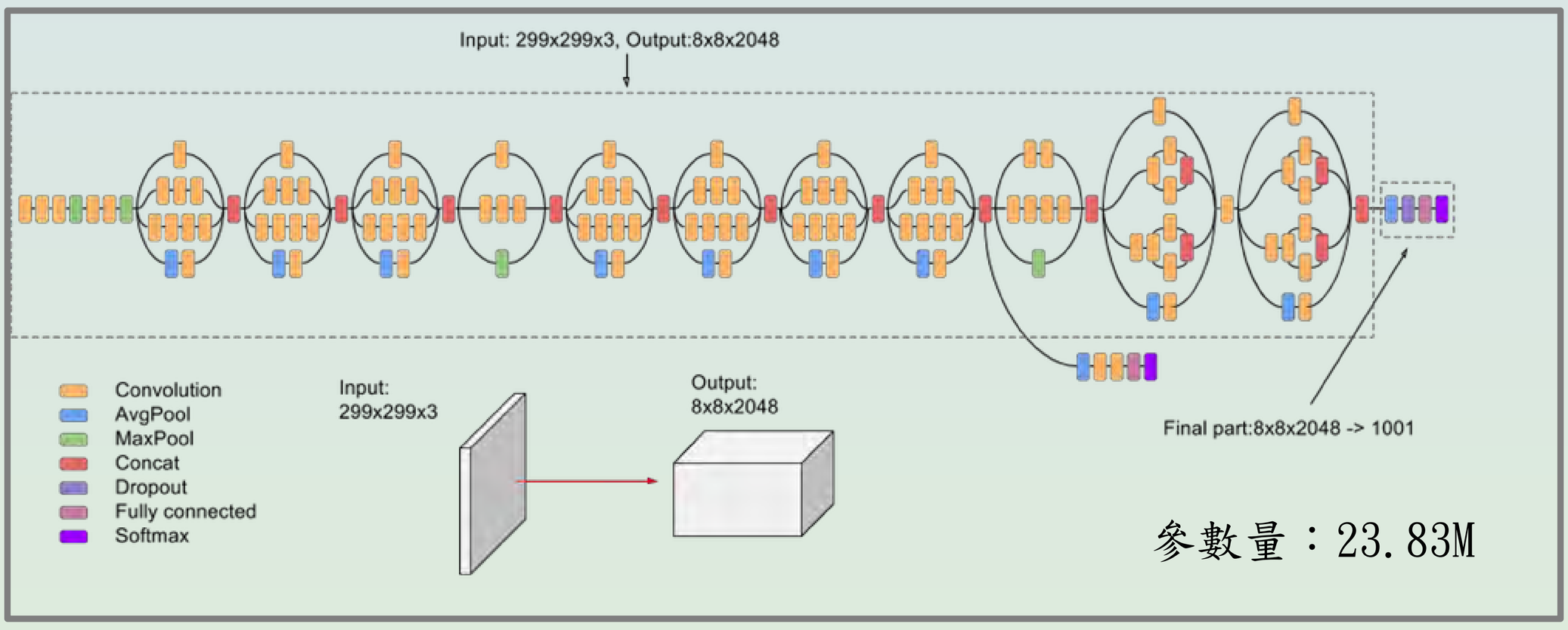


➢ 圖11
即時影像



柒、討論--影像辨識

一、使用模型：Inception V3



二、訓練方式：移轉學習

1. 經過了多次的試驗，最終採用「移轉學習」作為本系統的訓練方式。移轉學習為一透過引入他人訓練過的部分參數在自己的資料集上進行在訓練的方式。
2. 訓練資料數：每個類別各100張

三、辨識結果

1. 測試資料數：每個類別各10張
2. 測試結果：

	2人	3人	4人	5人
辨識類別	Justin/eagle	Justin/eagle/brian	Justin/eagle/brian/bear	Justin/eagle/brian/bear/chosen
測試acc	18/20=90%	27/30=90%	37/40=92.5%	40/50=80%

捌、未來展望及結論

Step1
- **增加靈活度**：利用在手機錄一段影像後傳至電腦並擷取影格，再加入訓練資料集來進行訓練，藉此增加新的訪客

Step2
- **增加全面性**：希望在辨識錯誤時，主人可以主動回報，並藉此將該判斷錯誤之照片加入訓練資料集，藉此增加辨識的全面性

Step3
- **增加可靠性**：透過學習新的演算法及蒐集更多資料來提升辨識的準確率，期待最終能達成臉即是鑰匙的目標

本次研究的智慧門禁系統與過往的傳統門禁系統有別：由樹莓派拍下訪客的照片，交給電腦進行影像辨識，將結果傳到使用者手中，再由他/她決定是否為其開門。 這樣的系統省去了很多過去繁瑣的過程，影像辨識加快了傳統需要透過門眼察看，或主人不在家時需要透過通話確認的步驟。結合手機應用程式，能夠即時通知使用者訪客的到來，清楚顯示在app中，讓使用者能一目了然，若使用者擔心影像辨識結果有誤，還能察看樹莓派拍下的照片做為確認。本次所製作的系統，提供了一種另類卻又不失安全性的門禁方式。