臺北市第 56 屆中小學科學展覽會

作品說明書封面

科　　別：電腦與資訊學科

組　　別：高級中等學校組

作品名稱：利用手部辨識模型實現遠端遙控

關 鍵 詞：機器學習、手部辨識、遠端遙控

編　　號：

摘要

在使用手機或是其他裝置的過程中，總有可能會遇到必須操作裝置，但卻在一個危險或不方便的情況，面對這種狀況時我們設計出一種新的方式來解決這種問題。本研究藉由 Python 模組中的 MediaPipe 來判斷出手部的節點位置，並且經過處理後儲存為資料，將所收集到的資料匯集成資料庫，利用資料庫輸入建立起模型訓練，訓練出模型後將其導入機器判斷使用者的手勢，而在成功的判斷出手勢後將其轉為相應的指令，以 Android Debug Bridge 為橋梁，將所要執行的指令輸入到控制裝置中，便完成我們一整道的流程。用手勢來控制裝置的好處在於不需要有一個固定的作業平台，隨時隨地都能進行操作，且裝置主體也僅需要一台樹莓派即可，這樣不僅輕便、方便且不需要太高的成本就能達到預想中的效果。

壹、前言

一、研究動機

根據統計，死傷車禍事後查出駕駛持用手機，二〇一六年五十六件，前年增至九十八件，去年九十五件，全國警方每年開罰逾兩萬件持用手機違規，相關死傷事故仍逐年上升，因此本組希望可以尋找出一套使汽機車駕駛也可以安全使用手機的方法，先前本組在網路上找資料時，查找到了利用 Arduino 套件組裝傳感器手套進行電腦手部辨識，再連結電腦進行輸入的相關研究，但，這項研究中所需要使用的許多組件價格十分昂貴，整隻手套價格甚至須達五萬多元，因此本組希望構思出成本更加低廉的方法。最後本組構想出，若利用手部辨識模型是否可以協助實現輕便且成本低廉手機遠端控制。

二、研究目的

(一) 蒐集訓練用手部節點資料，並且建立資料庫。

(二) 利用資料庫中資料訓練手部辨識模型。

(三) 連結裝置 Android 系統進行遠端遙控。

(四) 結合手部辨識模型結合樹梅派進行遠端遙控。

三、文獻回顧

(一) 目前擁有幫助在行車過程中使用電子設備的工具之缺陷

目前市面上有的工具：語音辨識系統、Arduino 機械手套：

1. 語音辨識系統：

語音辨識系統，顧名思義是可以透過人說出的話進行辨識，這樣的方法可以幫助閱讀障礙者更加清晰的溝通，但它也有幾項致命的缺點，首先，年齡和口音會阻礙系統識別語音的能力，再者，人類的語調變化和斷句可能會導致辨別的準確度降低，再加上各種不同語系的俚語、諺語，系統更需要大量資料庫來進行辨識，最後，語音辨識非常容易因為環境嘈雜而導致準確度下降，即便是語音辨識大家百度在吵雜的環境中也只能達到 81%的準確度（2014, Coates A., & Ng, A. Y.）。

2. Arduino 機械手套

這項工具可以運用在遠端控制機械臂、仿生機器人、小車等領域，雖然控制較精準，但因其設備昂貴，重量也可能會造成負擔，而且若手勢較為複雜，期也會無法辨識，還可能會因為手大小差別而導致無法適用。

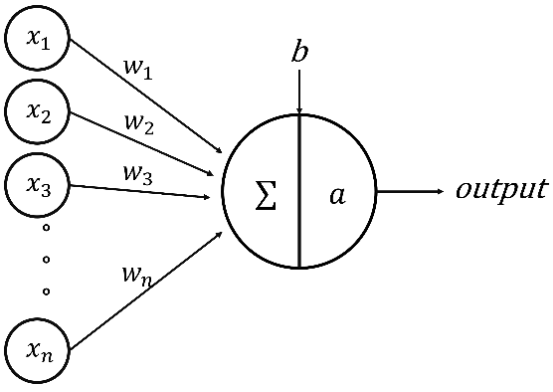
(二) 使用工具原理之探討

1. 神經網路

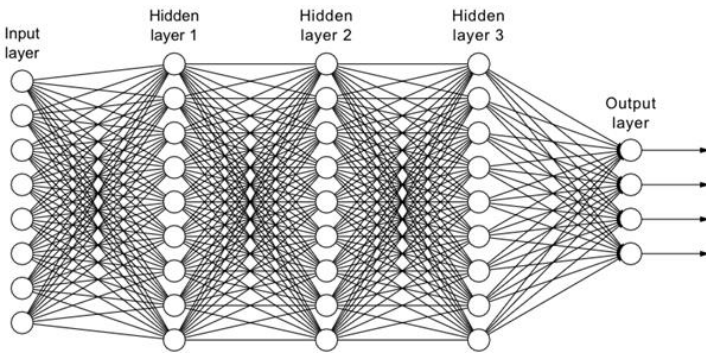
神經網路（Neural Network，NN）是一種模仿生物中樞神經網路的結構和功能的數學、計算模型，是一種進行機器學習的方法，其中的每個神經元（Neurons）皆會進行數學運算，而其運算皆可以表示成數學式，對於每一項輸入 x 進行加權 w 運算，再進行加總和偏差 b 修正，再透過激勵函數 a，最後輸出 $output$，而多層神經網路便是利用上一層輸出的 $output$ 做為下一層輸入 x 進行運算。

$$output = a\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

▲ 式一、單個神經元運算公式



▲ 圖一、單個神經元



▲ 圖二、多層神經網路

而運算出來的值便可以依照他所代表的意義進行資料判斷，以本研究為例（如式二），輸入為 21 個節點所組成的 21 維向量，而運算出來的輸出格式為不等維度的向量，每一個維度代表一種命令（手勢），若其值為此向輛中最大的值，那他代表的命令（手勢）將是這筆資料經過模型判斷出的答案。

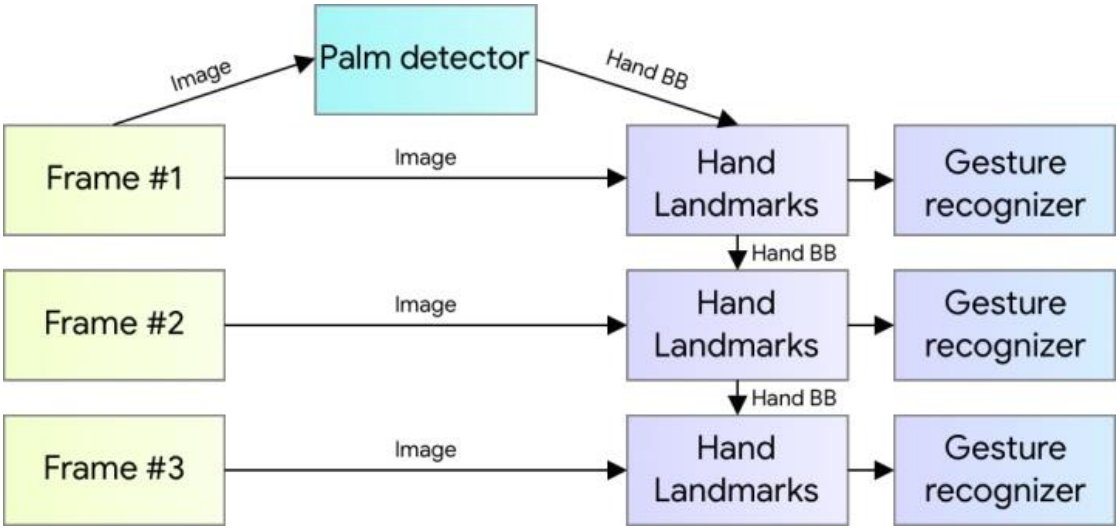
$$a(W[x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_{21}] + b) = [0 \quad 0 \quad \cdots \quad 0.2 \quad 0.7]$$

▲ 式二、本研究輸入輸出格式

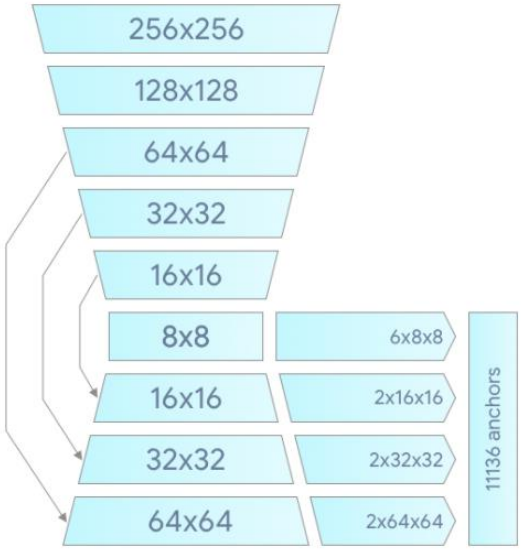
2. Mediapipe 框架

Mediapipe 框架是款由 Google 推出的開源機器學習管線框架，可以用來處理影像、聲音等時間序列資訊（Time Series Data），而最新整合進來的手部姿態辨識功能，提供了快速且精確的辨識品質，甚至能夠進行即時辨識。

首先，系統會先使用稱為 BlazePalm 的手掌偵測模型從圖像中鎖定手掌的區域，接著使用手部區域辨識模型在鎖定區域中標記關鍵 3D 定位點，最後則是使用手勢辨識系統判讀 21 個定位點、節點深度，左右手，根據關節的角度判斷每根指頭是伸直或彎曲，來判斷其精確手部動作（如圖四）。

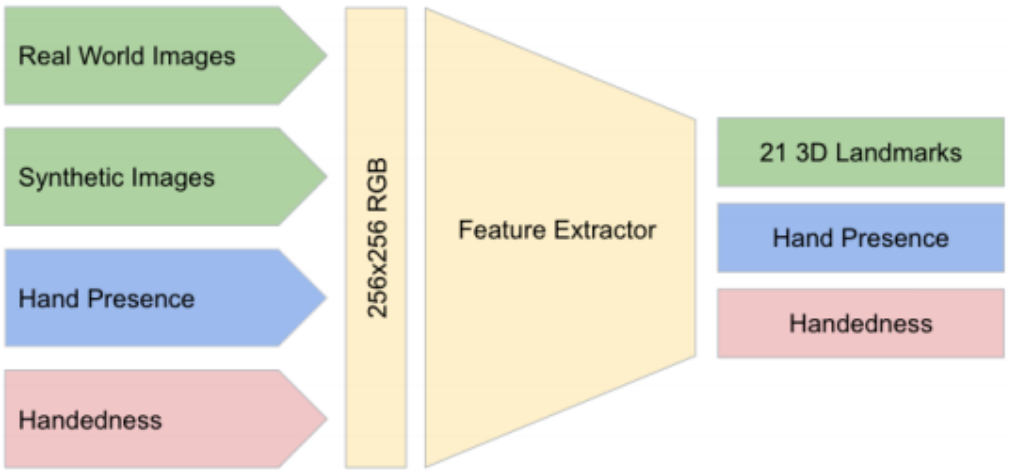


▲ 圖三、MediaPipe 框架



▲ 圖四、BlazePalm 手部偵測模型

BlazePalm 是一種類似於 FPN（Feature Pyramid Networks）的模型，利用小特徵來偵測大物件的結構設計。



▲ 圖五、手部節點標記模型架構

將影像及 BlazePalm 模型獲得隻手部位置資訊放入手部節點標記模型中，便會輸出 21 個手部節點位置、深淺、左右手。

貳、研究設備及器材

一、硬體

(一) Expert Center E5 AiO 24 (E5402)

CPU：Intel(R) Core(TM) i5-11500B CPU @ 3.30GHz 3.30 GHz

記憶體：8.0 GB

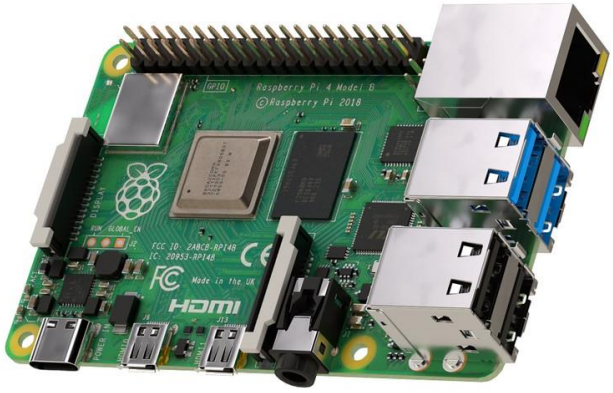
作業系統：Windows 11

(二) 樹梅派 Raspberry Pi 4 Model B

CPU：Broadcom BCM2711

記憶體：4GB

作業系統：Raspberry Pi OS



▲ 圖六、Raspberry Pi 4 Model B

二、軟體環境

(一) Python 3.10.7

(二) Visual Studio code

(三) OpenCV 4.6.0、MediaPipe 0.8.11、TensorFlow 2.9.2

(四) Android Debug Bridge 1.0.41

三、網頁工具

(一) Google Colab

參、研究過程或方法

一、訓練模型之資料收集與資料處理

(一) 手部資料擷取

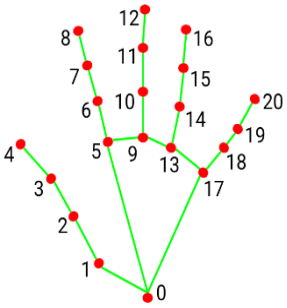
本組利用 OpenCV 加上 MediaPipe 擷取手部骨架節點的功能擷取 21 個手部節點，再接著進行以下兩者微調：

1. 正反面鏡像翻轉：

以第 0 個點為基準，將第 1 個點定位在第一象限，當第 1 個點的 x 值變為負值時，就將全部節點以 y 軸進行翻傳；同理，如果第 1 個點的 y 軸變為負值，則將所有點的 x 軸進行翻轉。

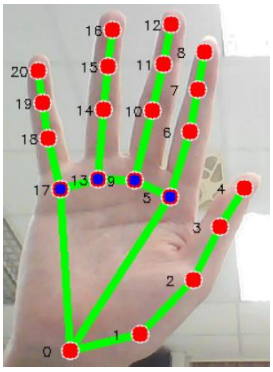
2. 手部比例調整：

以 0 ~ 5 的距離為準，將其距離訂為 1（由於距離較長），而其他節點的座標以其為基準，進行等比例的縮放來避免不同人之間由於手的大小不同而產生的誤差。



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

▲ 圖七、手部 21 個節點



▲ 圖八、本組擷取結果

(二) 資料存取

本組使用自行擷取的資料庫，並非是現成的資料數據庫，因此本組設計出一種方法：利用 Python 中 numpy 函式庫建立起專用的資料庫，記錄手部節點，最後用建立起的資料庫輸入訓練環境，使模型可以準確接收到資料。

```
if i == 0:
    x0 = xPos
    y0 = yPos
xPos = xPos - x0
yPos = y0 - yPos

x_turn = False
y_turn = False
```

▲ 圖九、紀錄節點 0 座標程式碼

```
if i == 1:
    if xPos < 0:
        x_turn = True
    if yPos < 0:
        y_turn = True
if x_turn == True:
    xPos = -xPos
if y_turn == True:
    yPos = -yPos
```

▲ 圖十、鏡像翻轉手部圖像程式碼

```
if i == 0:
    x5 = x5 - x0
    y5 = y5 - y0
    length_standard = sqrt(x5 * x5 + y5 * y5)
xPos_ch = (xPos / length_standard)
yPos_ch = (yPos / length_standard)
```

▲ 圖十一、手部比例調整程式碼

```
while True:
    ret, img = video.read()
    if ret:
        imgHeight = img.shape[0]
        imgWidth = img.shape[1]
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        result = hands.process(imgRGB)
        if result.multi_hand_landmarks:
            for handLms in result.multi_hand_landmarks:
                for i, lm in enumerate(handLms.landmark):
                    xPos = int(lm.x * imgWidth)
                    yPos = int(lm.y * imgHeight)
```

▲ 圖十二、擷取節點程式碼節錄

二、模型建構與模型訓練

(一) 神經網路建構

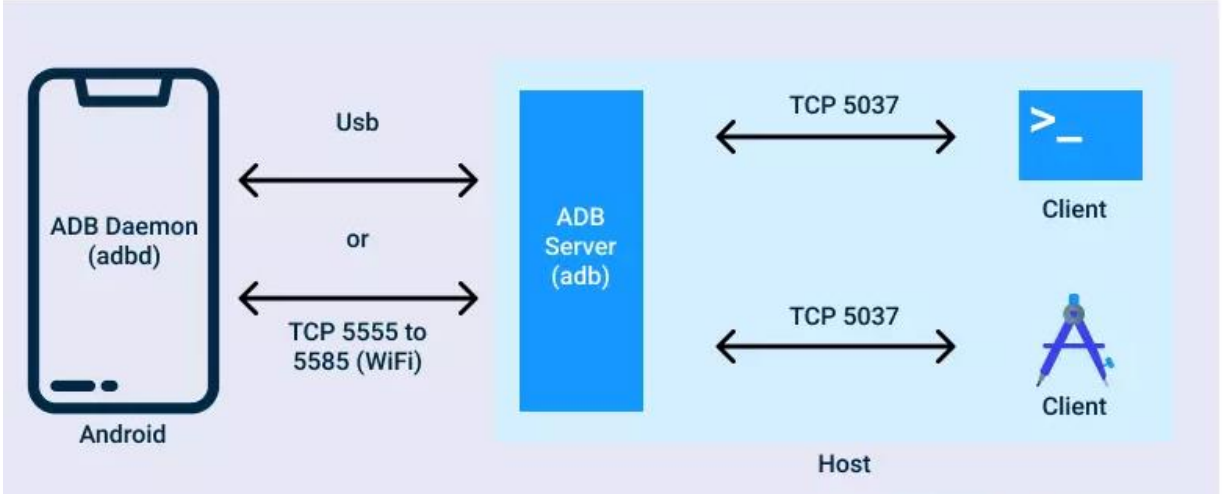
本組在獲得手部節點資料後，開始進行機器學習，因本組資料型態及模型性質的原因，本使用深度學習模型（Deep Neural Network）加上監督式學習（Supervised Learning）進行建構。

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 1000)              43000

 dense_1 (Dense)              (None, 1000)              1001000

 dense_2 (Dense)              (None, 1000)              1001000

 dense_3 (Dense)              (None, 26)                26026

=================================================================
Total params: 2,071,026
Trainable params: 2,071,026
Non-trainable params: 0
_________________________________________________________________
```

▲ 圖十三、神經網路架構

三、使用 ADB（Android Debug Bridge）作為連接橋梁

在模型辨識出手勢並且發出訊號後，我們將會擷取模型發出的訊號，將擷取到的訊號放入程式中，在取得相應指令後，以 ADB 這個橋梁做為媒介分別從兩端連接，將電腦與手機連至伺服器，接著將模型的輸出作為訊號，透過發送至 ADB 伺服器，Wi-Fi 連接，使伺服器接收到模型判斷的訊號，最後從伺服器傳送指令遠端操控手機，就能在手機上做出各種操作。



▲ 圖十四、裝置連接橋梁示意圖

四、結合樹梅派（Raspberry Pi）實現遠端遙控

(一) 設置環境：

將樹梅派接起螢幕、滑鼠、鍵盤，連結鏡頭，安裝 OpenCV、Mediapipe、Tensorflow、ADB 等框架。

(二) 下載程式與模組：

從雲端下載事先準備好的手部辨識程式與訓練完成的手部辨識模型，將其組合，進行手部動作辨識。

(三) 連結裝置與執行程式

使用 abd 的指令將樹莓派與指定電子裝置連結，再執行程式，使其能夠辨識手勢並操控手機

肆、研究結果

一、蒐集訓練用手部節點資料，並且建立資料庫

(一) 資料蒐集、存儲、連結：本組研究所需資料並未有先例，因此在資料蒐集及連結方面需要找到方法，本組使用 OpenCV 加上 MediaPipe 套件來進行手部資料擷取，蒐集每個手勢 3000 筆資料，共 24 種手勢，總計 72000 筆資料，利用鏡頭進行輸入，最終以鏡頭中心為基準點，輸出 21 個節點的座標。而輸出的做標本組使用 numpy 的 array 進行存儲，再放入.npz 檔中，最後使用 load 函式輸入進入 Colab 進行模型訓練。

```
x_train[rem, i, 0] = xPos_ch
x_train[rem, i, 1] = yPos_ch
y_train = np.zeros((100, 26))
for i in y_train:
    y_train[..., gestrue_num] = 1
np.savez(file_name, x = x_train, y = y_train)
```

▲ 圖十五、存儲資料程式碼片段

二、利用資料庫中資料訓練手部辨識模型

(一) 梯度消失（Gradient Vanish）：本組剛開始訓練模型的時候是使用 Sigmoid 函數作為激勵函數，而我們發現訓練到達 60%後，會變得非常困難移動，而我們發這這種問題應該是源自於梯度消失，這樣的問題是因為 Sigmoid 求導後會發現，x 值越大梯度越小，最終導致無法訓練，因此本組改用 ReLu 做為激勵函數來進行訓練。

$$S(x) = \frac{1}{1+e^{-x}}$$

▲ 式三、Sigmoid

$$S'(x) = S(x)\cdot(1-S(x))$$

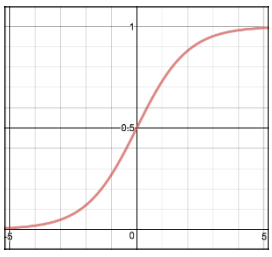
▲ 式四、Sigmoid'

$$R(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

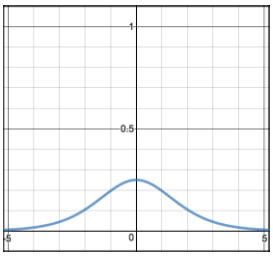
▲ 式五、ReLu

$$R'(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$

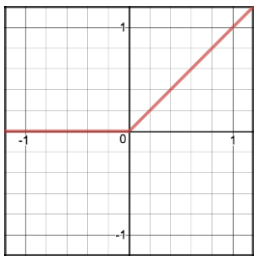
▲ 式六、ReLu'



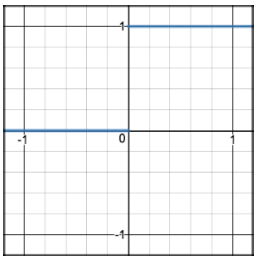
▲ 圖十六、Sigmoid 函數



▲ 圖十七、Sigmoid 導數

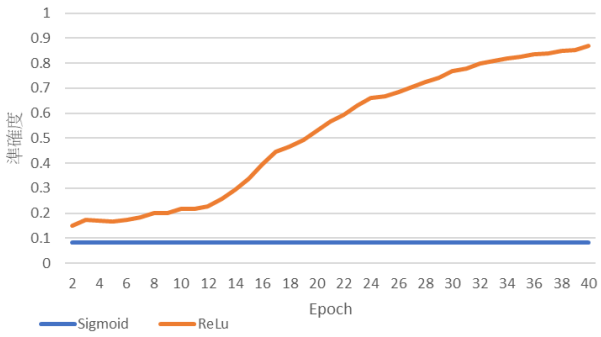


▲ 圖十八、ReLu 函數

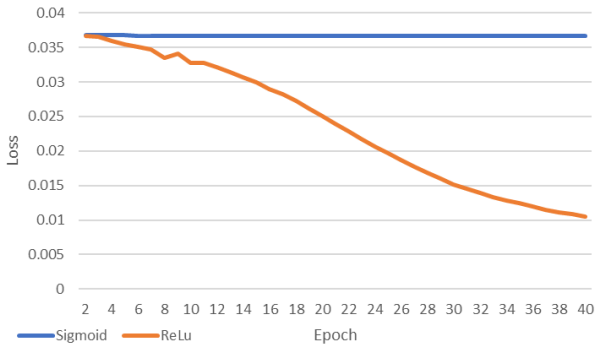


▲ 圖十九、ReLu 導數

以下為實際改善後結果（藍色為 Sigmoid、橘色為 ReLu）：



▲ 圖二十、激勵函數對準確度作圖



▲ 圖二十一、激勵函數對 Loss 作圖

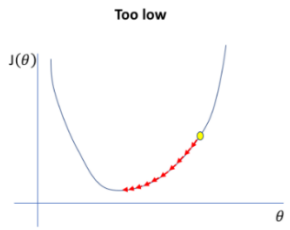
(二) 學習速度（Learning Rate）：在深度學習中，我們會用「學習率（Learning Rate）」來控制學習的速度，而學習率會影響到訓練出的模型好壞，圖中所展現的函數為損失函數（Loss function），損失函數的值越低代表訓練出來的效果越好而本組使用均方誤差 MSE（如式七）作為損失函數，而根據訓練更新公式（如式八），如果學習率 α 太低（如圖十八），會導致訓練的效率太低，容易落入局部最小值，難以達到理想的效果；而如果學習率太高（如圖二十），很容易就會越過理想的區域，導致學習的結果不準確。

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

▲ 式七、均方誤差

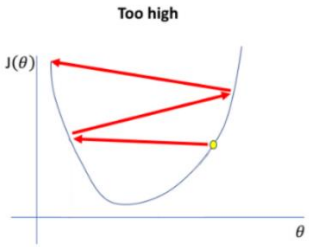
$$x_{i+1} = x_i - \eta\frac{\partial L}{\partial x_i} = x_i - \eta\nabla L(x_i)$$

▲ 式八、訓練資料更新公式



▲ 圖二十二、學習率過低

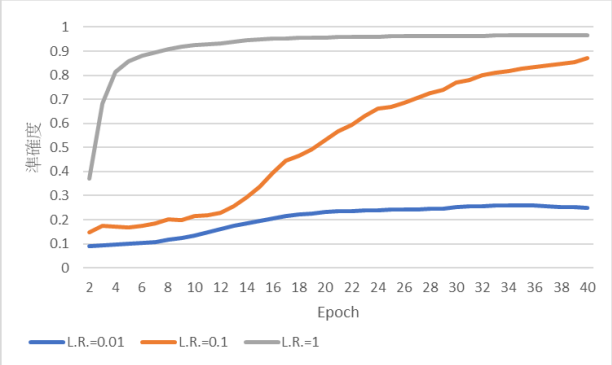
▲ 圖二十三、理想學習率



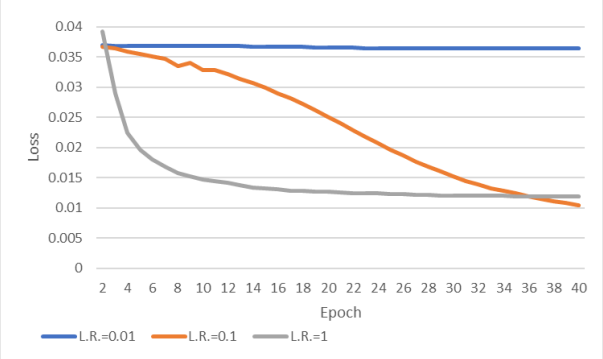
▲ 圖二十四、學習率過高

以下為實際改善後結果（灰色：L.R.=1、藍色：L.R.=0.1、橘色：L.R.=0.01）：

本組在設定過程中分別嘗試了不同的學習率，而由圖二十二中可以看出當學習率設為 1 時會由於學習率設定的值過高而無法達到最好的效果，而當學習率設為 0.01 時則會產生學習率過低的狀況，導致損失函數中的值並無明顯下降，而將值設為 0.1 是一個比較適合的值，因此在往後本組擴增資料庫的研究中應選擇 0.1 作為學習速率。

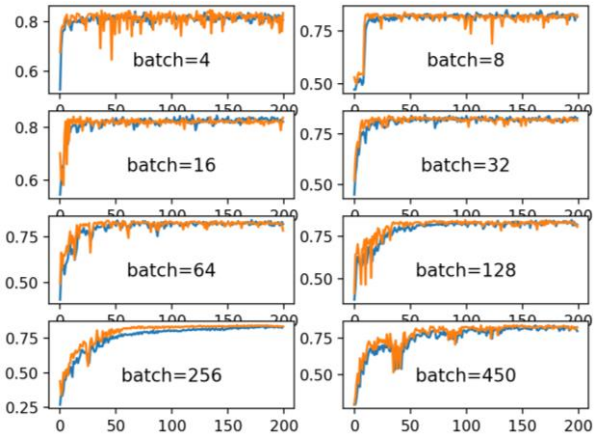


▲ 圖二十五、Learning Rate 對準確度作圖

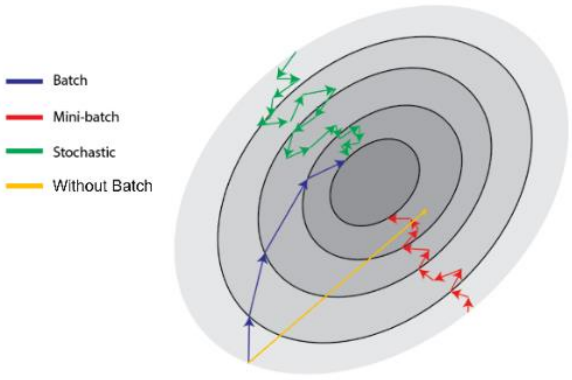


▲ 圖二十六、Learning Rate 對 Loss 作圖

(三) Batch size、Epochs：在訓練過程中會分批次的將資料送入函式中進行學習，Epochs 代表的是我們要將資料重複訓練幾次（幾輪），而 Batch size 則是我們一次要送幾筆資料進去學習，舉例而言，當我們有 100 筆資料，我們把 Epochs 設為 20，Batch size 設為 25，在這種情況下我們一次會送 25 筆資料進行學習，在重複送入資料 4 次後稱為一輪，並且重複以上步驟 20 次，我們可以藉由這種分批學習的方式來慢慢逼近我們最理想狀態，如果一次將資料全部置入，資料庫中雜訊對準確率會產生過大的影響，從而導致結果的偏差。

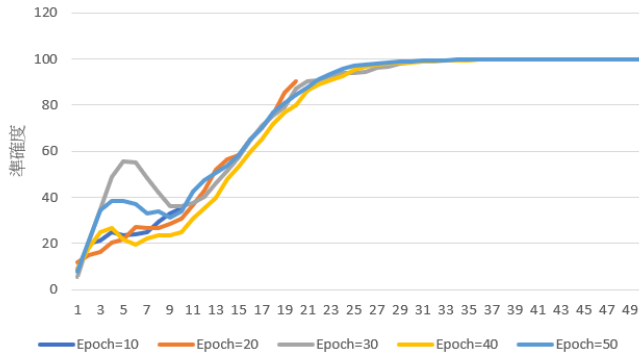


▲ 圖二十七、Batch size 對學習影響

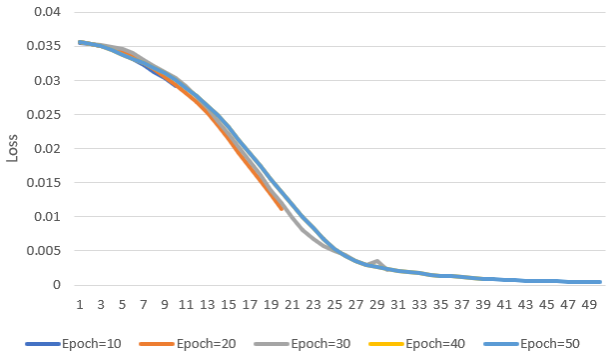


▲ 圖二十八、使用 Batch 前後

以下為本組針對 Epoch 對模型訓練影響之作圖：



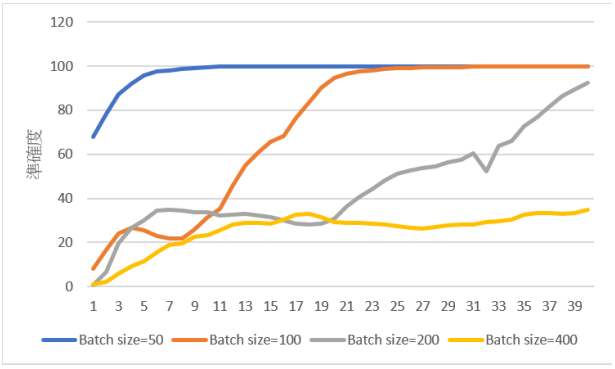
▲ 圖二十九、Epoch 對準確度作圖



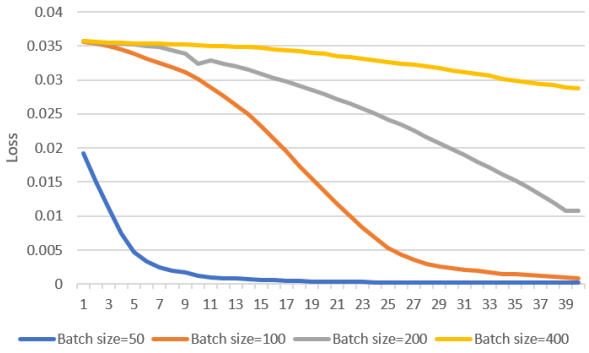
▲ 圖三十、Epoch 對 Loss 作圖

從圖表中可以觀察到兩件事，從 Loss 函數方面來講，若 Epoch 值不夠會造成函數值下降量不足的問題，導致訓練出來的模型判斷效果不佳。再來，若依據準確度作圖來觀察，一樣是模型訓練不完全，導致訓練出來的模型判斷效果不佳，不過，本組還發現了另外一個問題，在模型參數更新時，可能會有一開始進入局部最小值的情況，若 Epoch 值不夠大，即使 Learning Rate 夠大，也會有無法走出局部最小值的情況。

以下為本組針對 Epoch 對模型訓練影響之作圖：



▲ 圖三十一、Batch size 對準確度作圖



▲ 圖三十二、Batch size 對 Loss 作圖

從圖表中可以觀察到，對於 Batch size 來說當值太大時，模型便容易因 Loss 無法到達全域最小值而導致模型無法最佳化，而若 Batch size 太小時，其容易受單筆資料影響，訓練過多次，而導致其下降至局部最小值，無法訓練出最佳化模型。

(四) 最終神經網路訓練

1. Activation function：ReLu
2. Batch size：100
3. Epoch：40
4. Learning Rate：0.1
5. 模型準確度：89.17%

```
model = Sequential()

model.add(Dense(1000, input_dim=42, activation='relu'))
model.add(Dense(1000, activation='relu'))
model.add(Dense(1000, activation='relu'))
model.add(Dense(26, activation='softmax'))
```

▲ 圖三十三、神經網路建構程式碼

三、連結裝置 Android 系統進行遠端遙控

以下為幾組測試用功能：

(一) 基礎滑動：

包含上下左右四個方向的滑動，可結合其他功能做使用。

```
upswipe = '500 300 500 600'
downswipe = '500 600 500 300'
leftswipe = '100 800 1000 800'
rightswipe = '1000 800 100 800'

if (predict_max == 14):
    device.shell(f'input swipe {upswipe}')
if (predict_max == 15):
    device.shell(f'input swipe {downswipe}')
if (predict_max == 16):
    device.shell(f'input swipe {leftswipe}')
if (predict_max == 17):
    device.shell(f'input swipe {rightswipe}')
```

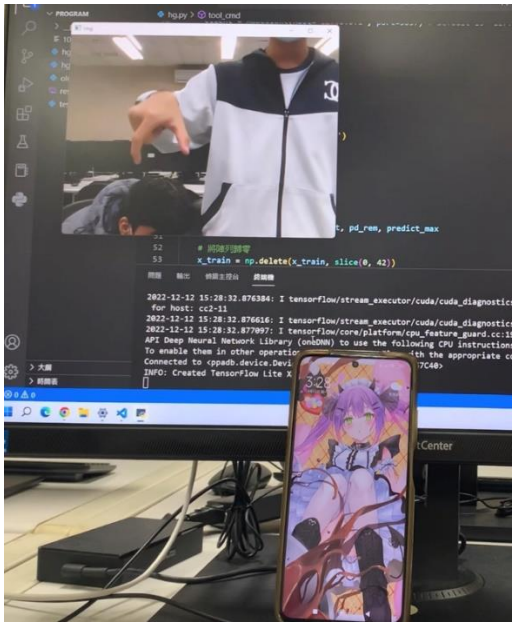
▲ 圖三十四、滑動功能程式碼片段

(二) APP 應用程式開啟：

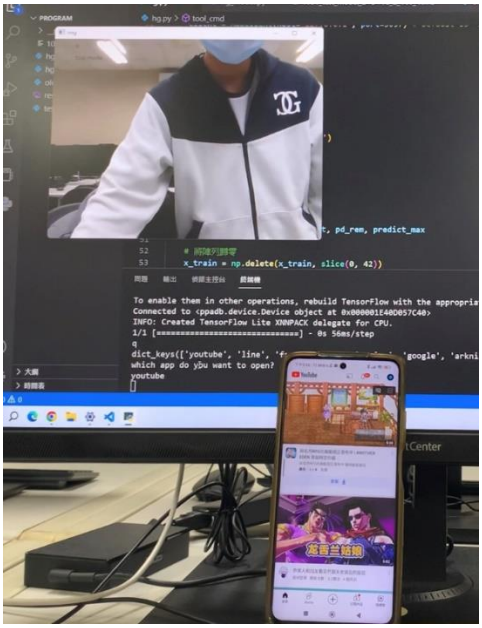
在名為 command 的 dictionary 中加入想要使用的 APP，便可以使用手勢打開 APP。

```
def m_appOpen():
    command = {
      "name1" : 'link1',
      "name2" : 'link2',
      "name3" : 'link3',
      "name4" : 'link4',
      "name5" : 'link5'}
    print(command.keys())
    Input = input('which app do you want to open?\n')
    device.shell(f'am start -n {command[Input]}')
```

▲ 圖三十五、APP 應用程式開啟功能程式碼片段

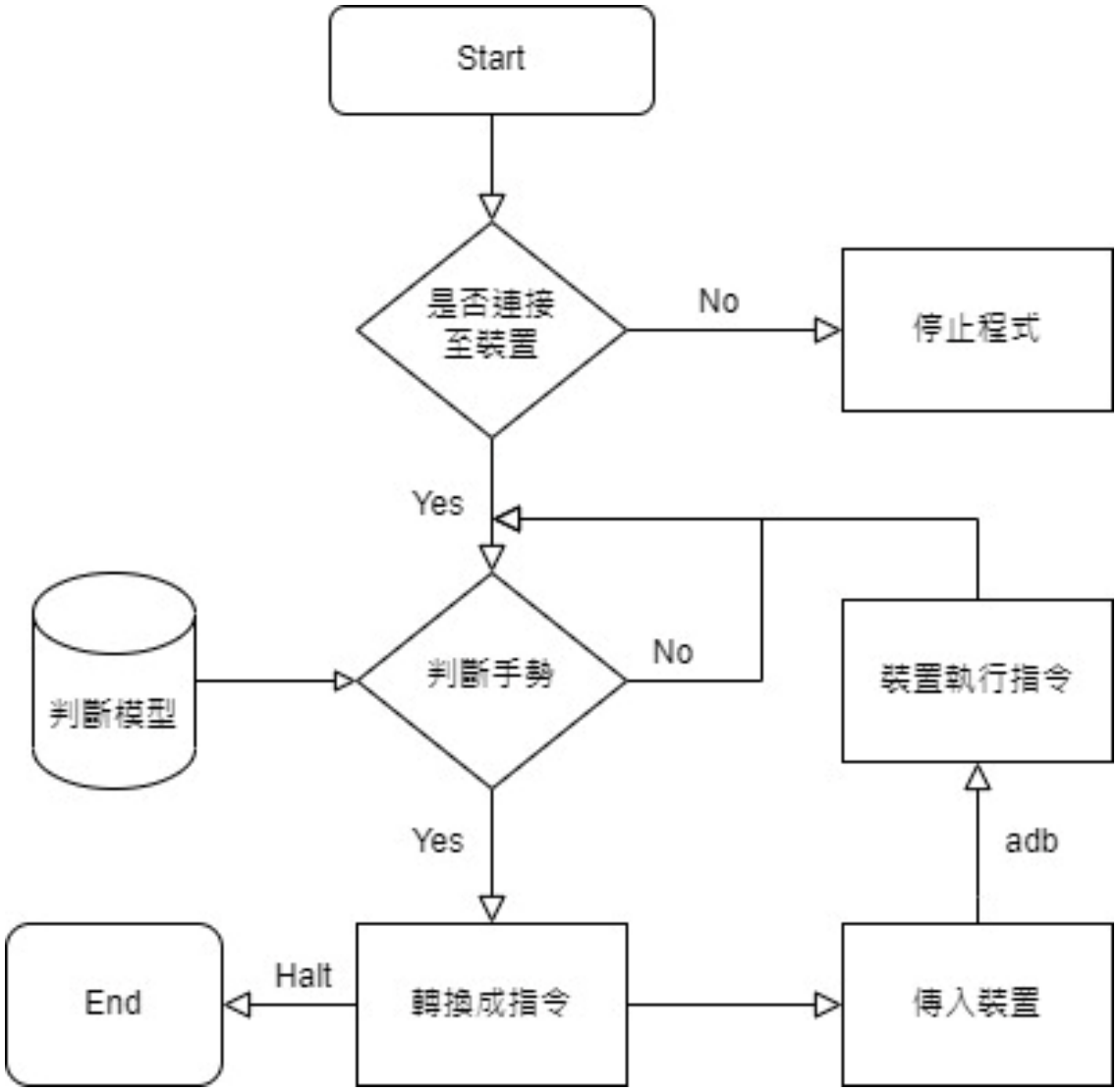


▲ 圖三十六、APP 應用程式開啟



▲ 圖三十七、APP 應用程式開啟

四、結合手部辨識模型進行遠端遙控



▲ 圖三十八、裝置運行流程圖

1. 執行程式
2. 確認是否連接至裝置
    i. 是：繼續執行程式
    ii. 否：停止執行程式
3. 偵測手部節點並判斷手勢（藉由導入的判斷模型）
    i. 判斷失敗：辨識失敗，重新判斷
    ii. 判斷成功：執行指令
4. 結束指令

伍、討論

一、探討機器學習模型資料蒐集問題：

本研究所需資料並無在網路上並無先例，因此在蒐集資料上本組利用 numpy 來建立一套資料庫，而從研究結果來看，本組建立的資料庫是成功的。

二、探討機器學習模型參數：

如結果所示，機器學習參數包含四項：激勵函數、學習速率、Batch Size、Epoch，本組在充分了解各項參數所代表意涵後得出以下結果：

(一) 使用 ReLu 函數作為激勵數可消除梯度消失的問題，進而解決模型 Loss 值停滯不嚴的問題。

(二) 找到適當的學習速度使模型訓練獲得進一步的效率提升。

(三) 找到適當的 Epoch 及 Batch Size，使模型準確率上限增加

三、探討裝置連結效益：

裝置連接方式分兩種：USB 連接與 WIFI 連接。USB 連接僅需啟動 ADB 伺服器，並將裝置與電腦都連接到同一個 Wi-Fi，即可連接裝置；而 Wi-Fi 連接除了上述兩步操作外，還需要輸入裝置 IP 進行連接。不過亦有其他因素導致各有優異之處，說明如下：雖說 USB 連接可獲得較高的傳輸速率，但使用起來卻不方便，因此本組利用 Wi-Fi 可達到無線連接的效果，來進行橋樑搭建。

四、未來展望：

(一) 資料蒐集、資料庫建構：即便本組此次在蒐集資料上取得了一定的成果，但，對於一個機器學習的模型來說這樣的資料數量是遠遠不夠的，因此本組希望可以在蒐集更多的資料，重新進行訓練，對手勢辨識模型進行優化。

(二) 手勢辨識模型參數優化：本組此次僅針對激勵函數、學習速度、Batch Size、Epoch，四項參數進行優化，但在機器學習的領域中，還有多方式可以對此模型進行優化，例如：損失函數（Loss Function）的選擇、隱藏層（Hidden Layer）的數量選擇，甚至是使用不同的模型建構方式等。本組希望可以針對這幾項變因對模型進行優化。

(三) 未來應用：本組希望可以利用樹梅派（Raspberry Pi）等成本較低的微型電腦，導入本組的模型，進行手部辨識實現遠端控制，進而達到降低汽機車駕駛因使用手機而造成危險的情況。

陸、結論

一、本研究藉由 TensorFlow 訓練出一套有效的手勢辨識模型，能夠有效地對使用者所做出的手勢進行辨識，從而做出進一步的指示及對裝置的操作。

二、藉由相同的訓練方式能夠辨識出更多的手勢，增加指令的多樣性以應對不同的環境，且可以讓不同的手勢下達專門的指令甚至是一系列流程的操作，以達到客製化的效果。

三、使用 ADB 作為與 Android 系統連接的橋樑能夠有效地對裝置下達指令，且能夠支援 Wi-Fi 的連接功能，僅需要連線至相同 Wi-Fi 便能實現有效的遠程控制。

四、未來希望可以搭配樹梅派作為主機即可進行操作，不僅造價低廉同時還能夠兼具實用性，且基於樹梅派輕便的特性能夠更加方便的將裝置帶往各種環境進行使用。

五、當處於特殊環境下難以操作裝置時（例：醫師必須處於安靜環境下開刀），無法手動操作也無法語音控制操作設備，都可以藉由此裝置來進行控制，有效的解決問題；如若是有身體行動不便者也可以藉由簡單的手勢來代替實際的操作。

柒、參考文獻資料

[1] Gary Bradski, Adrian Kaehle（2008）。Learning OpenCV: Computer Vision with the OpenCV Library.

[2] Aisha Ali-Gombe, Irfan Ahmed, Golden G. Richard, Vassil Roussev(2016). AspectDroid: Android App Analysis System. DA MOTA Alexandre（2020）。

[3] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung Chuo-Ling, Chang Matthias Grundmann（2020）MediaPipe Hands: On-device Real-time Hand Tracking.

[4] 劉明山（2009）。基於視覺之手部追蹤與手勢辨識之研究。

[5] DA MOTA Alexandre（2020）。Création dun bot automatisé basé surlapprentissage par reinforcement Travail de Bachelor réalisé en vue de.

[6] 陳柏元、王孟輝（2012）。應用網路攝影機和可拓理論設計一套低成本之手部辨識系統。

[7] Siddharth S. Rautaray（2012）。Real Time Hand Gesture Recognition System for Dynamic Applications。https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3702844

[8] 許常志（2010）。手部姿態辨識之研究。

[9] Google Developer（2022 年 8 月 24 日）。Android Debug Bridge（ADB）使用手冊。https://developer.android.com/studio/command-line/adb?hl=zh-tw

[10] Michael Burton（2015）. Android App Development For Dummies.

[11] 蔡炎龍（2018 年 10 月 19 日）。磨課師課程－蔡炎龍－Python 實現人工智慧。政大開放式課程影音網。https://ctld.video.nccu.edu.tw/km/1172

[12] 李文恩（2017 年 4 月 4 日）。Hayo 神奇遙控器，用手勢就能操作多種電器。https://www.techbang.com/posts/49972-hayo-magic-remote-control-gestures-will-be-able-to-operate-a-variety-of-electrical-appliances

[13] AxiomQ（2022）. What Are the Advantages and Disadvantages of Speech Recognition? https://axiomq.com/blog/what-are-the-advantages-and-disadvantages-of-speech-recognition/

[14] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger R, Satheesh S, Sengupta S, Coates A., & Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition.