蛇類影像辨識 App

Snake Image Recognition App

指導老師：李國川教授

組員：楊善富、呂鑒峰

國立聯合大學 資訊工程學系

苗栗市南勢里聯大 2 號

gclee@nuu.edu.tw

{U0624047, U0624048}@o365.nuu.edu.tw

摘要

台灣山區常有蛇類出沒，但人們大多不熟悉毒蛇的特徵。因此我們開發這款 App，幫助登山客快速辨識蛇的種類及其毒性以迴避危險情況，或是能在被咬時能第一時間告知醫護人員蛇的品種。

我們以 TensorFlow Lite 建立並訓練影像辨識的 AI 模型，大幅縮小了整個程式的體積，並以 Android Studio 作為整合 AI 及開發 App 其他功能的平台。

關鍵詞：蛇、影像辨識、AI、App。

Abstract

Snakes are often found in Taiwan's mountainous areas, but most people are not familiar with the characteristics of poisonous snakes Therefore, we developed this App, to help climbers quickly identify the type of snake and its toxicity to avoid dangerous situations, or to inform the medical staff of the snake species as soon as it bit.

We build and train an image recognition model with TensorFlow Lite, which greatly reduced the size of the entire App. And we use Android Studio as a platform to integrate AI and develop other functions of App.

Keyword: snack, image recognition, AI, App

一、研究動機與目標

(一) 動機

台灣山區風景優美，吸引許多遊客登山。但山上蛇類眾多，其中也有劇毒且攻擊性強的毒蛇。若沒有事先做足功課就難以辨認蛇是否有毒。此時若能快速的辨識出蛇的品種，便可大大降低風險，被咬時也能與救護人員反映蛇的品種，因此想開發出操作簡單且能快速辨識各種蛇類的圖像辨識 AI。

(二) 目標

考慮到使用此圖像辨識的背景環境，我們選擇手機作為運作軟體的裝置，開發出行動裝置的 App，一來可以使用手機的相機鏡頭獲得目標圖像，二來手機輕巧，方便攜帶，在危急時刻較能快速開啟程式。

主要目標為開發出的 App 能有影像辨識的 AI，可以快速辨識出目標蛇的品種。

但由於山區的網路連線品質不一，若將辨識部分的程式放在雲端伺服器，在危急時可能無法及時獲得辨識結果，因此影像辨識的部分必須在 App 內完成。

再考慮到手機的硬體性能比一般的電腦低，因此需要更小、更便於計算的模型。

二、使用的軟體及開發平台

我們訓練 AI 的環境是由 Google 所提供的平台 Colaboratory，此平台可免費提供 GPU 資源，且環境更容易建置。我們以 TensorFlow Lite 的函式庫訓練 AI，再以 Android Studio 整合 AI 及 App 開發其他功能的平台。

(一) Google Colaboratory

簡稱為 Colab，是 Google 提供的免費服務，只要有 Google 帳號就能使用。當使用瀏覽器進入 Colab 時，會進入到名為「Colab 筆記本」的互動式環境。

Colab 筆記本環境類似於 Jupyter，能直接撰寫及執行 Python，但 Colab 不用進行任何設定，且運算用的 GPU 也是免費提供的。

除此之外還能連結 Google 雲端硬碟，從中讀取檔案或將檔案寫入，亦能與其它 Google 帳號共用 Colab 筆記本，可讓他人加入註解或協助編輯程式。

(二) TensorFlow/TensorFlow Lite

TensorFlow 是一款由 Google Brain 團隊開發的開源函式庫，用於各種感知和語言理解任務的機器學習。

而 TensorFlow Lite 則是為了能直接在行動裝置、嵌入式裝置、物聯網裝置上執行 TensorFlow 模型的工具，分為直譯器及轉換工具兩部分。直譯器著重於精簡及快速，並確保有最小的負載及執行延遲。而轉換工具則是能將 TensorFlow 模型或 Keras 模型轉換成方便直譯器使用的格式，並透過最佳化來降低檔案大小及提高效能。

TensorFlow Lite 預設使用的模型為 EfficientNet Lite0，為卷積式神經網路。與原本的 EfficientNet 相比，雖然同樣是卷積式神經網路，但 EfficientNet Lite 使用 ReLU6 取代 Swish，並捨棄 SE-block (squeeze-and-excitation)，使整個模型更加適合移動裝置。

由於 TensorFlow Lite 不必透過網路來回傳送資料，因此可以降低網路連線造成的延遲及耗電，再加上資料無須離開裝置，能確保資料的隱私性。

(三) Android Studio

Android Studio 是由 Google 開發的整合式開發環境，使用的程式語言為 Java 或 Kotlin，並可在 Windows、macOS、Linux 等作業系統上執行。

開發出的應用程式可以藉由傳輸線直接安裝在 Android 作業系統的手機上，或是建立 Android 系統的虛擬機執行。亦可輸出成 apk 檔，便於發布在網路上供人安裝。

三、系統功能開發

(一) 影像辨識

首先我們在 Colab 環境中以 pip 安裝 tflite-model-maker 套件，並載入 numpy、TensorFlow 及 TensorFlow Lite 相關函式庫，再加上用於連接 Google 雲端硬碟的 drive 函式庫。載入的函式庫如下圖 1。

```
[ ] !pip install -q tflite-model-maker
```

```
[ ] from tflite_model_maker import image_classifier
    from tflite_model_maker.image_classifier import DataLoader

    import tensorflow as tf
    assert tf.__version__.startswith('2')

    import matplotlib.pyplot as plt
    import numpy as np

    from google.colab import drive
```

圖 1. 使用的套件及函式庫

而訓練 AI 用的圖片，我們選擇上傳至雲端硬碟，再與 Colab 進行連結，讓 Colab 能直接讀取。我們先在雲端硬碟裡建立了一個名為 /content/drive/snake 目錄，並在該目錄底下建立各種蛇類的子目錄，再將所有照片放入對應目錄中。最後只需要將訓練 AI 時讀取檔案的路徑設定為 /content/drive/snack，訓練時便能直接讀取雲端硬碟中的圖片。

接下來進行訓練圖片的資料預處理，使用 TensorFlow Lite 的函數從雲端硬碟中獲得資料，一方面讀取/snake 之下子目錄的數量及名稱，作為輸出層的神經元數量及輸出結果；另一方面將二維圖像轉化成一維資料，作為輸入層的神經元輸入。之後便是建立模型及訓練。

由於真實的蛇不易取得，且有安全上的風險，我們選擇使用擬真的塑膠蛇作為

影像辨識的訓練材料，以此模擬真實情況並測試辨識模型是否可行。我們總共使用三種不同的塑膠蛇，分別為大眼鏡蛇、小眼鏡蛇、雨傘節，每種蛇各有 3000 張圖片進行訓練。訓練耗費的時間約為 1 小時。

```
model = image_classifier.create(train_data)
```

```
INFO:tensorflow:Retraining the models...
Model: "sequential"
```

Layer (type)	Output Shape	Param #
hub_keras_layer_v1v2 (HubKerasLayerV1V2)	(None, 1280)	3413024
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 3)	3843

```
Total params: 3,416,867
Trainable params: 3,843
Non-trainable params: 3,413,024
```

圖 2. 建立影像辨識模型

```
Epoch 1/5
235/235 [==============================] - 1325s 6s/step - loss: 0.3675 - accuracy: 0.9770
Epoch 2/5
235/235 [==============================] - 500s 2s/step - loss: 0.3193 - accuracy: 0.9989
Epoch 3/5
235/235 [==============================] - 501s 2s/step - loss: 0.3153 - accuracy: 0.9996
Epoch 4/5
235/235 [==============================] - 513s 2s/step - loss: 0.3128 - accuracy: 0.9993
Epoch 5/5
235/235 [==============================] - 505s 2s/step - loss: 0.3112 - accuracy: 0.9997
```

```
loss, accuracy = model.evaluate(test_data)
```

```
27/27 [==============================] - 198s 5s/step - loss: 0.3032 - accuracy: 1.0000
```

圖 3. 訓練結果

訓練結束後，將訓練完成的模型輸出成 tflite 檔案，並存於個人雲端硬碟的資料夾中，我們再從雲端硬碟下載檔案。訓練完成的模型大小約為 4,000KB，與其他裝置用的模型相比確實小了許多。

(二) 手機 App

我們以 TensorFlow Lite 提供的 App 範例為基礎，並加入我們訓練的模型。由於範例程式中僅開啟相機進行辨識，沒辦法對畫面進行縮放，也沒有開啟手電筒的功能，因此在程式碼中加入這兩項功能。

```
val cameraControl = camera.cameraControl
val cameraInfo = camera.cameraInfo
val listener = object : ScaleGestureDetector.SimpleOnScaleGestureListener() {
    override fun onScale(detector: ScaleGestureDetector): Boolean {
        val currentZoomRatio = cameraInfo.zoomState.value?.zoomRatio ?: 0F
        val delta = detector.scaleFactor
        cameraControl.setZoomRatio(currentZoomRatio * delta)
        return true
    }
}
val scaleGestureDetector = ScaleGestureDetector( context: this, listener)
viewFinder.setOnTouchListener { _, event ->
    scaleGestureDetector.onTouchEvent(event)
    return@setOnTouchListener true
}
```

圖 4. 縮放功能及照明功能

四、App 實際成果

啟動手機 App 後，會自動開啟手電筒照明，並即時對相機鏡頭拍攝的畫面進行影像辨識，其每次辨識所花費的時間短的令人感到驚奇，每秒都能有十數次的辨識結果。

圖 5. App 執行畫面

在不改變相機縮放倍率，僅控制手機位置調整目標在畫面佔比的情況下，距離約 15 公分時，辨識結果有 95%以上將正確

答案列為第一項。只有背景為黑白相間時，會有機率將其他兩種蛇誤判為雨傘節，推測為這部分的訓練樣本不足導致。

若與目標距離很遠，僅以控制畫面縮放倍率的方式調整目標大小，則準確度會依實際距離而逐漸下降。

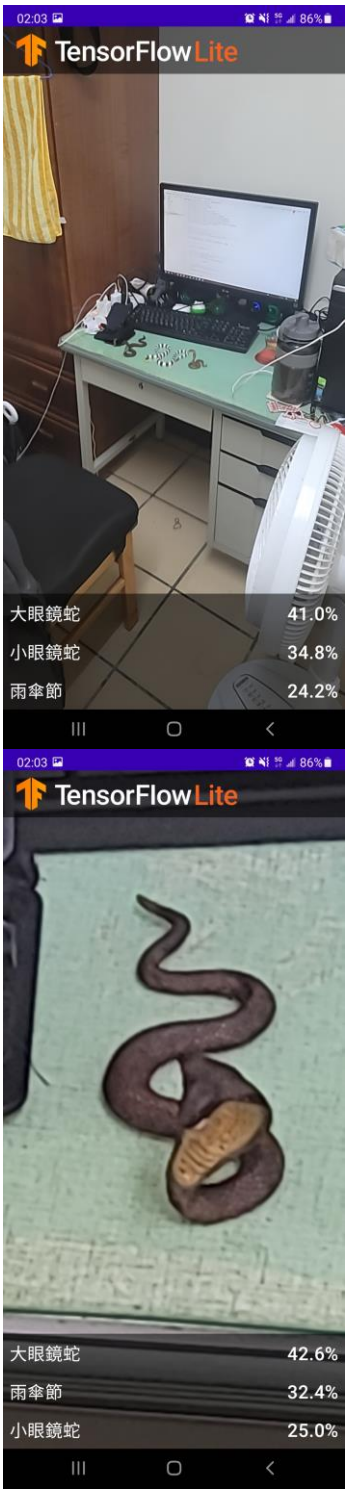

圖 6. 將遠處目標放大後進行辨識

光線不足時，由於有程式自動開啟的閃光燈當作光源，因此辨識結果與光線充足時相差不大，遠距離時亦然。

圖 7. 僅使用手機閃光燈作為光源辨識

圖 8. 僅使用手機閃光燈作為光源，將遠處目標放大後進行辨識

當然，上述測試結果可能會因手機不同而有不同的辨識成功率，舉凡鏡頭像素及閃光燈亮度等都會影響到辨識結果。

五、結論

本次專題透過 TensorFlow Lite 的輕量化模型，成功在手機上實作影像辨識。雖然在部分情況下準確度因訓練樣本不足而有待加強，但大多數時候的準確度及辨識所需的時間令我們感到十分驚艷。

學習過程中接觸到的輕量化辨識模型技術及操作手機攝影鏡頭的知識著實讓我們成長許多。甚至中間曾因 TensorFlow 更新過訓練模型而重新訓練模型，並將整個 App 打掉重做，但這整個過程仍讓我們學習到許多新知。期望這個模型能有真正的蛇類作為訓練素材，相信必定能在關鍵時刻派上用場。

參考文獻

〔1〕Image classification | TensorFlow Lite, 2021

https://www.tensorflow.org/lite/examples/image_classification/overview?hl=zh-tw

〔2〕Recognize Flowers with TensorFlow Lite on Android, 2021

https://codelabs.developers.google.com/codelabs/recognize-flowers-with-tensorflow-on-android/index.html?hl=zh-tw#0

〔3〕Android CameraX: Control and Query the Camera. | Husayn Hakeem | ProAndroidDev, 2021

https://proandroiddev.com/android-camerax-tap-to-focus-pinch-to-zoom-zoom-slider-eb88f3aa6fc6

〔4〕分類 — EfficientNet-lite - 22 12 – Medium

https://gino6178.medium.com/%E5%88%86%E9%A1%9E-efficientnet-lite-deb3598c0d58