5.5 使用 Swift 在 iOS 中使用簡易語音辨識模型

我們在第 2 章就已經用 Swift 做過一個 iOS app，並在其中加入了 TensorFlow pod。現在請新建一個 Swift app 並搭配上一節所建置的 TensorFlow iOS 函式庫，並在這個 Swift app 中使用語音指令模型：

1. 在 Xcode 中新建一個 Single View iOS 專案，設定方式與上一節的步驟 1、2 完全相同，但程式語言請選擇 Swift。

2. 點選 **File | New | File ...**，接著選擇 **Objective-C File**。名稱請輸入 `RunInference`，系統會詢問你 **"Would you like to configure an Objective-C bridging header?"** 按一下 **Create Bridging Header** 選項，把原本的檔名 `RunInference.m` 改名為 `RunInfence.mm`，因為之後要把 C、C++ 與 Objective -C 程式碼混搭起來進行聲音錄製後處理與辨識。這個 Swift app 中還是會用到 Objective-C，這是因為如果要從 Swift 來呼叫 TensorFlow C++ 程式碼的話，需要一個 Objective-C 類別作為 C++ 程式碼的封裝。

3. 建立一個名為 `RunInference.h` 的標頭檔，並加入以下內容：

```
@interface RunInference_Wrapper : NSObject
- (NSString *)run_inference_wrapper:(NSString*)recorderFilePath;
@end
```

你的 Xcode 設定畫面應該會如圖 5.5：

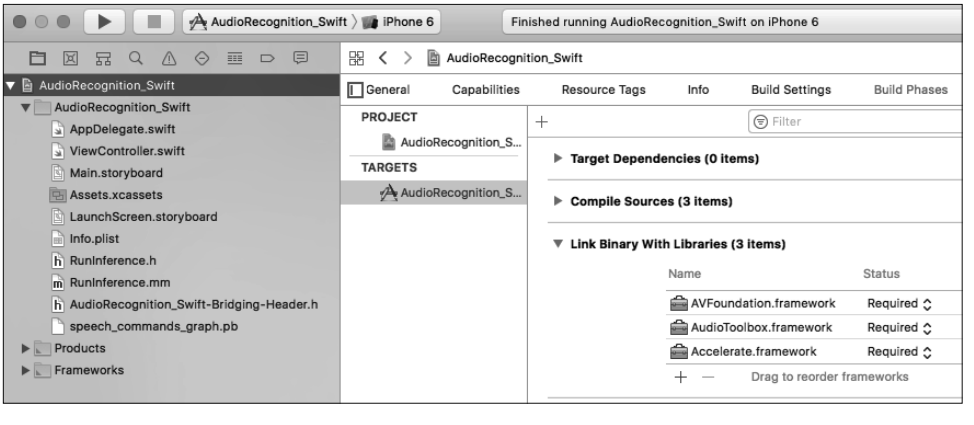

圖 5.5　使用 Swift 語言的 iOS 專案

4. 開啟 ViewController.swift，在 import UIKit 之後加入這一段：

```
import AVFoundation

let _lbl = UILabel()
let _btn = UIButton(type: .system)
var _recorderFilePath: String!
```

這會讓 ViewController 看起來像這樣（用來定義 _btn 與 _lbl 的 NSLayoutConstraint，以及呼叫 addConstraint 的程式碼在此省略）：

```
class ViewController: UIViewController, AVAudioRecorderDelegate {
    var audioRecorder: AVAudioRecorder!
override func viewDidLoad() {
    super.viewDidLoad()
    _btn.translatesAutoresizingMaskIntoConstraints = false
    _btn.titleLabel?.font = UIFont.systemFont(ofSize:32)
    _btn.setTitle("Start", for: .normal)
    self.view.addSubview(_btn)
    _btn.addTarget(self, action:#selector(btnTapped), for:
.touchUpInside)

    _lbl.translatesAutoresizingMaskIntoConstraints = false
self.view.addSubview(_lbl)
```

5. 加入按鈕點擊處理器，並於其中先要求錄音的使用者權限：

```
@objc func btnTapped() {
    _lbl.text = "..."
    _btn.setTitle("Listening...", for: .normal)
    AVAudioSession.sharedInstance().requestRecordPermission () {
        [unowned self] allowed in
        if allowed {
            print("mic allowed")
        } else {
            print("denied by user")
            return
        }
    }
```

接著建立一個 AudioSession 實例，將其類別設定為 record 並設為啟用狀態，作法如之前的 Objective-C 版本一樣：

```
let audioSession = AVAudioSession.sharedInstance()

do {
    try audioSession.setCategory(AVAudioSessionCategoryRecord)
    try audioSession.setActive(true)
} catch {
    print("recording exception")
    return
}
```

定義 AVAudioRecorder 會用到的相關設定：

```
let settings = [
    AVFormatIDKey: Int(kAudioFormatLinearPCM),
    AVSampleRateKey: 16000,
    AVNumberOfChannelsKey: 1,
    AVLinearPCMBitDepthKey: 16,
    AVLinearPCMIsBigEndianKey: false,
    AVLinearPCMIsFloatKey: false,
    AVEncoderAudioQualityKey: AVAudioQuality.high.rawValue
    ] as [String : Any]
```

設定儲存錄音的檔案路徑、建立一個 AVAudioRecorder 實例、設定其委派方法並開始錄音 1 秒鐘：

```
do {
    _recorderFilePath =
NSHomeDirectory().stringByAppendingPathComponent(path:
"tmp").stringByAppendingPathComponent(path: "recorded_file.wav")
    audioRecorder = try AVAudioRecorder(url: NSURL.fileURL(withPath: _
recorderFilePath), settings: settings)
    audioRecorder.delegate = self
    audioRecorder.record(forDuration: 1)
} catch let error {
    print("error:" + error.localizedDescription)
}
```

7

使用 CNN 及 LSTM 做繪圖辨識

前一章，我們見識到了使用結合 CNN 以及 LSTM RNN 的深度學習模型可以對一張影像產生自然語言描述的強大力量。如果把深度學習的 AI 當新的電力一樣來使用，可以期望在許多不同領域中看到這種混合神經網路模型的應用。相對於幫圖片加上說明這樣嚴肅的應用主題是什麼呢？一個有趣的繪圖 app 看起來是個不錯的選擇，像是 Quick Draw（`https://quickdraw.withgoogle.com`，在 `https://quickdraw.withgoogle.com/data` 可以看到有趣的範例）。它是使用了 345 個種類、5000 萬張圖畫訓練出的模型，並將新圖畫歸類到這些類別中。而且有 TensorFlow 官方教學（`https://www.tensorflow.org/tutorials/recurrent_quickdraw`）介紹如何建立這樣的模型，來幫助我們快速入門。

這項在 iOS 和 Android app 上使用此教學建立模型的任務提供了一個絕佳的機會讓你：

- 加深對模型的正確輸入和輸出節點名稱的理解，讓我們可以適當地準備適用於行動裝置 app 的模型。
- 使用其他方法來修復 iOS 中新模型的載入和推論錯誤。

- 首次為 Android 建立自定義的 TensorFlow 原生函式庫，以修復 Android 中的新模型載入和預測錯誤。
- 認識更多有關如何提供 TensorFlow 模型預期格式的輸入資料，以及如何在 iOS 和 Android 中獲取和處理其輸出的範例。

此外，在處理所有繁瑣但重要的細節，使模型可以像變魔術一樣漂亮的分類繪圖的過程中，你將在 iOS 和 Android 設備上享受塗鴉的樂趣。

本章內容如下：

- 繪圖分類的運作原理
- 訓練、預測、以及準備繪圖分類模型
- 在 iOS 中使用繪圖分類模型
- 在 Android 中使用繪圖分類模型

7.1 繪圖分類的運作原理

TensorFlow（`https://www.tensorflow.org/tutorials/recurrent_quickdraw`）中內建的圖形分類模型會先將使用者的繪圖輸入以點的清單來表示，然後將正規化輸入轉換為連續點之間增量的 tensor，以及有關每個點是否為新筆劃的開始的資訊。然後經過幾個卷積層和 LSTM 層、以及最後一層 softmax 層處理，對使用者的繪圖進行分類，如圖 7.1 所示：

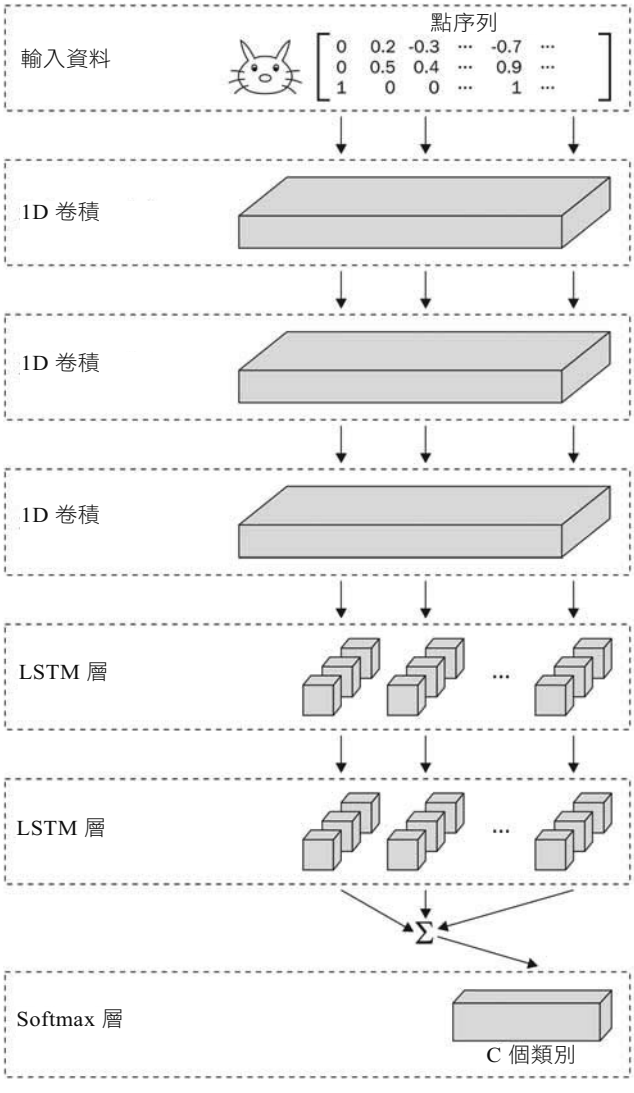

圖 7.1　繪圖分類模型

與接受 2D 圖像輸入的 2D 卷積 API `tf.layers.conv2d` 不同，此處像繪圖這樣的時間性卷積採用了 1D 卷積 API `tf.layers.conv1d`。預設情況下，在圖形分類模型中使用三個 1D 卷積層，每個層具有 48、64 和 96 個卷積核，其長度分別為 5、5 和 3。在卷積層之後，將建立 3 個 LSTM 層，每層具有 128 個正向

9

使用 GAN 生成和強化影像

從 2012 年，深度學習開始大受歡迎以來，有人認為沒有什麼新想法比 Ian Goodfellow 在 2014 年的論文**生成對抗網路**（**Generative Adversarial Networks**）（https://arxiv.org/abs/1406.2661）中介紹的**生成對抗網路**（Generative Adversarial Network, GAN）更有趣或有前途了。實際上，Facebook AI 的研發總監、也是開創性的深度學習研究人員之一的 Yann LeCun 稱 GAN 和對抗訓練為「過去 10 年以來，機器學習領域中最有趣的想法」。因此，我們一定要在這裡介紹它、了解 GAN 為什麼如此令人興奮、以及如何建立 GAN 模型並在 iOS 和 Android 上執行。

本章將從概述 GAN 是什麼，它如何運作以及為什麼具有如此巨大的潛力開始。接著，將介紹兩個 GAN 模型：一個可用於生成類似真人寫的手寫數字的基礎 GAN 模型，與另一個可將低解析度影像增強為高解析度影像的進階 GAN 模型。我們將示範如何在 Python 和 TensorFlow 中建立和訓練此類模型，以及如何為行動裝置部署及準備模型。我們提供附有完整原始程式碼的 iOS 和 Android 應用程式，它們將使用這些模型來生成手寫數字並增強影像。在本章節結束後，你應該準備好進一步探索各種基於 GAN 的模型，或者開始建立自己的模型，並了解如何在行動裝置的應用程式中執行它們。

本章涵蓋內容如下：

- 何謂 GAN，為什麼要使用它
- 使用 TensorFlow 建立和訓練 GAN 模型
- 在 iOS 中使用 GAN 模型
- 在 Android 中使用 GAN 模型

9.1 何謂 GAN，為什麼要使用它

GAN 是學習生成類似於真實資料或訓練集中資料的神經網路。GAN 的關鍵想法是讓生成器網路和鑑別器網路相互競爭：生成器試圖生成看起來像真實資料的資料，而鑑別器試圖分辨生成的資料是真的（已知的真實資料）或假的（由生成器生成的資料）。生成器和鑑別器是一起訓練的，在訓練過程中，生成器學習如何生成看起來越來越像真實資料的資料，而鑑別器則學習辨別真實資料與假資料。用生成器的輸出作為鑑別器的輸入時，生成器透過試著讓鑑別器輸出的真實資料機率盡可能接近 1.0 來學習；而鑑別器則透過試著達成以下兩個目標來學習：

- 以生成器的輸出作為輸入時，讓輸出的真實資料機率盡可能接近 0.0，這恰好與生成器的目標相反
- 以真實資料作為輸入時，讓輸出的真實資料機率盡可能接近 1.0

info

下一節，你將看到與生成器網路、鑑別器網路及其訓練過程描述符合的詳細程式片段。如果你想了解更多關於 GAN 的知識，除了這裡的簡述之外，你還可以在 YouTube 上搜尋 ***Introduction to GANs***，並觀看在 2016 年 NIPS（Neural Information Processing Systems）會議和 2017 年的 ICCV（International Conference on Computer Vision）會議上 Ian Goodfellow 的 GAN 介紹和教學影片。在 YouTube 上，總共可以找到 7 支 2016 年 NIPS 對抗訓練工作坊影片和 12 支 2017 年 ICCV 的 GAN 教學影片。

在生成器和鑑別器這兩者的競爭下，GAN 是一個尋求兩個對手之間保持平衡的系統。如果兩者都有無限大的能力並能做到訓練最佳化，就可以達到納許均衡（以 1994 年諾貝爾經濟學獎得主約翰．納許以及電影《美麗境界》而命名），代表任一方都無法藉由改變自身策略來提升獲利的穩定狀態。這正好是我們的生成器確實可生成栩栩如生的資料，而鑑別器無法辨識其真偽的狀態相符。

info

如果你有興趣詳細了解何謂納許均衡，在 Google 上搜尋 ***khan academy nash equilibrium*** 可以找到兩支 Sal Khan 的有趣影片。納許均衡的維基百科頁面和「納許均衡是什麼以及它為何重要」（What is the Nash equilibrium and why does it matter?）這篇經濟學人雜誌文章也很不錯，了解 GAN 背後的基礎直覺和想法將幫助你更加了解其為何具有巨大潛力。

`https://www.economist.com/blogs/economist-explains/2016/09/economist-explains-economics`

生成器能夠生成看起來像真實資料的潛力，意味著有很多各式各樣出色的應用程式能使用 GAN 來達成，例如：

- 用低解析度影像生成高解析度影像
- 影像修復（修復消失或毀損的影像）
- 翻譯影像（例如，將線稿草圖變成照片，或者在人臉上添加或移除像是眼鏡之類的物件）
- 從文字生成影像（與在第 6 章介紹的 Text2Image 相反）
- 撰寫看起來像真實新聞的新聞文章
- 生成與訓練集中的聲波相似的聲波

基本上，GAN 具有從隨機輸入的內容生成逼真的影像、文字或聲波資料的潛力。如果你具有一組包含原始資料和目標資料的訓練集，則 GAN 還可從與原始資料相似的輸入中生成與目標資料相似的資料集。這是 GAN 模型中生成器

以下程式碼顯示如何在 newckpt 目錄中來指定佔位符值、執行 GAN 模型並儲存生成器的輸出以及檢查點檔案：

```
if a.mode == "test":
    from scipy import misc
    image = misc.imread("ww.png").reshape(1, 256, 256, 3)
    image = (image / 255.0) * 2 - 1
    result = sess.run(output_image, feed_dict={image_feed:image})
    misc.imsave("result.png", result.reshape(256, 256, 3))
    saver.save(sess, "newckpt/pix2pix")
```

圖 9.1 顯示了原始測試影像、它的模糊版本以及經過訓練的 GAN 模型的生成器輸出的影像結果。結果雖然不理想，但是 GAN 模型確實具有更高的解析度而且沒有模糊效果：

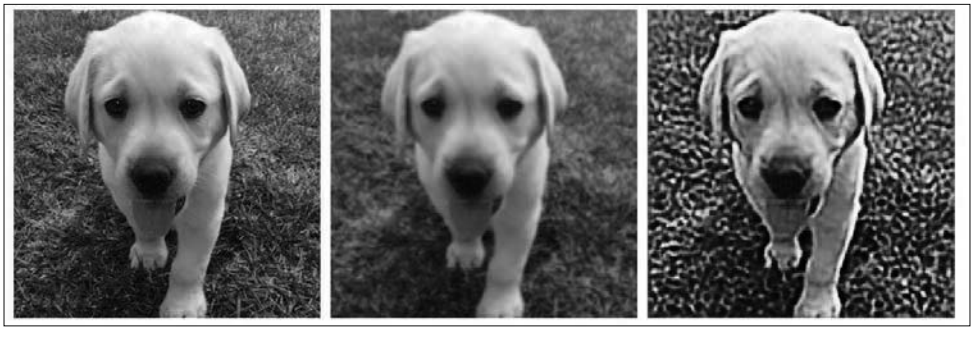

圖 9.1　原始影像、模糊化影像、生成影像

10. 現在，將 newckpt 目錄複製到 /tmp，並凍結模型如下：

```
python tensorflow/python/tools/freeze_graph.py \
--input_meta_graph=/tmp/newckpt/pix2pix.meta \
--input_checkpoint=/tmp/newckpt/pix2pix \
--output_graph=/tmp/newckpt/pix2pix.pb \
--output_node_names="generator_1/deprocess/truediv" \
--input_binary=true
```

11. 生成的 pix2pix.pb 模型檔案很大，約為 217 MB，在將其放到 iOS 或 Android 裝置上時會毀損或導致記憶體不足（**Out of Memory, OOM**）錯誤。

現在你的 Xcode 畫面應該類似圖 9.2：

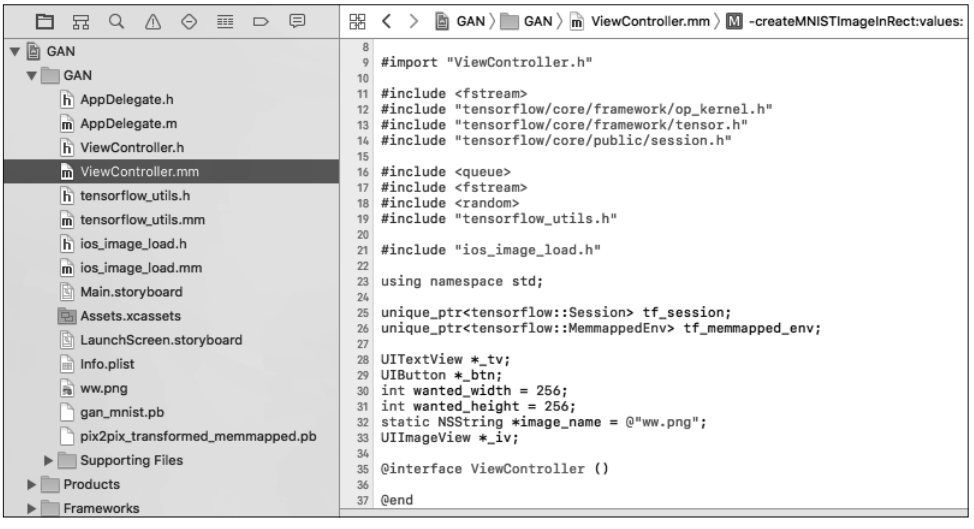

圖 9.2　在 Xcode 中顯示 GAN 應用程式

在此將建立一個按鈕，在點擊該按鈕時會請使用者選擇生成數字模型或是提高影像解析度模型這兩者其中之一：

```
- (IBAction)btnTapped:(id)sender {
    UIAlertAction* mnist = [UIAlertAction actionWithTitle:@"Generate
Digits" style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
        _iv.image = NULL;
        dispatch_async(dispatch_get_global_queue(0, 0), ^{
            NSArray *arrayGreyscaleValues = [self runMNISTModel];
            dispatch_async(dispatch_get_main_queue(), ^{
                UIImage *imgDigit = [self createMNISTImageInRect:_iv.frame
values:arrayGreyscaleValues];
                _iv.image = imgDigit;
            }];
        });
    });
    UIAlertAction* pix2pix = [UIAlertAction actionWithTitle:@"Enhance
Image" style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
        _iv.image = [UIImage imageNamed:image_name];
        dispatch_async(dispatch_get_global_queue(0, 0), ^{
```

這些數字看起來很像人類親手寫的數字，都是在訓練了基礎 GAN 模型之後完成的。如果你回頭查看執行該訓練的程式碼，然後暫停一下，想一想 GAN 整體的工作方式，以及生成器和鑑別器如何相互競爭並嘗試達到穩定的納許均衡狀態，也就是生成器可以產生出鑑別器無法成功分辨的栩栩如生假資料，你應該更能體會 GAN 多麼令人驚奇。

現在，請選擇 **Enhance Image** 選項，你將看到如圖 9.4 的結果，該結果與圖 9.1 中的 Python 測試程式碼生成的結果相同：

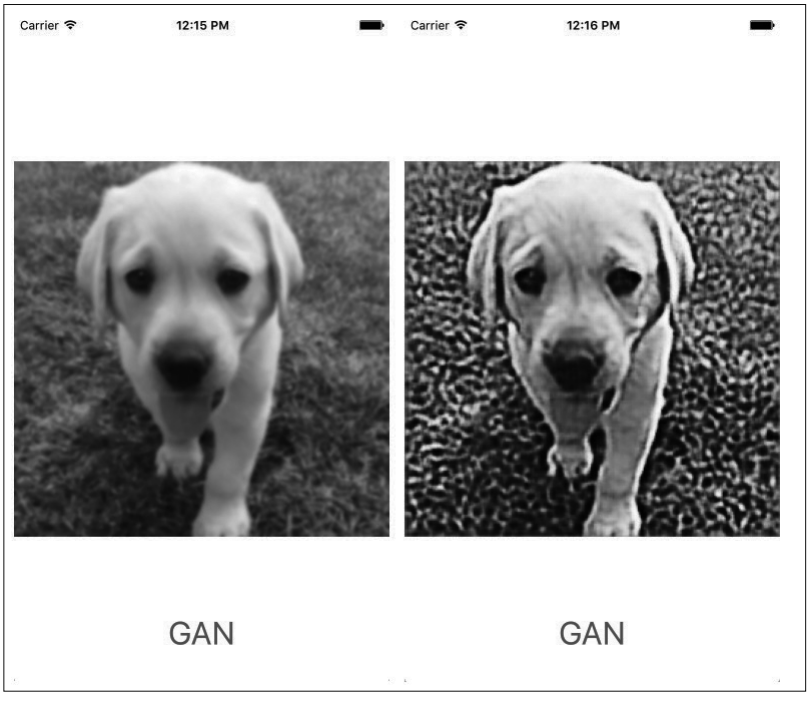

圖 9.4　在 iOS 上的原始模糊影像以及提高解析度後的影像

你知道的，是時候將我們的愛獻給 Android 了。