前言

我們正在經歷人工智慧的復興，每個人和他們週遭的人都想要參與這個行動。這大概就是您正在翻閱這本書的原因。坊間已經有了無數的深度學習書籍了，所以您大概會問一個合理的問題：本書為何還會問市呢？我們只需要一秒就能回答這個問題。

自 2013 年起我們開始了深度學習的旅程（藉由在 Microsoft、NVIDIA、Amazon 及 Square 這些公司製作產品）並見證了這個環境的劇烈變化。不斷進化的研究成果貢獻良多，不過缺乏成熟的工具則是我們必須面對的現實。

從社群的成長與學習中我們注意到，對於要將學術研究轉換為所有使用者都可用的產品這件事來說，經常缺少明確的指引。終究使用者總是會處在瀏覽器、智慧型手機，或者邊緣裝置前的某處。這過程常常涉及了無數的破解與實驗、部落格搜尋、GitHub 討論串、Stack Overflow 的回覆、寄給套件作者的電子郵件以取得內行人才懂的知識，還有偶爾會出現的「原來如此！」時刻。即使坊間的書籍試圖聚焦於理論或工具的運用，但我們從現有的書中能學到的最多只是建立虛構的範例。

要彌平理論與實務的鴻溝，一開始我們以演講方式來將人工智慧從研究帶到人群中，並聚焦於實際應用。這些演講被組織成引人入勝的範例，並依不同的技巧能力（從業餘愛好者到 Google 等級的工程師）以及在產品上部署深度學習的勞心程度來調整複雜度。我們發現不論初學者還是專家都從這些演講中找到有價值之處。

還好隨著時間進展，環境對使用者是友善的，同時也出現了更多的工具。像是 Fast.ai 以及 DeepLearning.ai 這樣的優良線上資源，讓訓練人工智慧模型這件事比以往容易多了。使用像是 TensorFlow 和 PyTorch 這類深度學習架構來教導基礎知識的書也出現在市面上。即使如此，介於理論與產品間的深淵還是沒有被提及。我們想要彌補其間的差距，因此出現了您正在看的書。

碁峰 www.gotop.com.tw

藉由使用友善的語言以及立即可用的有趣電腦視覺專案，本書會從不需要任何機器學習與人工智慧背景知識的簡單分類器開始，逐步的增加複雜度、改善準確度與速度、調整至百萬使用者規模、部署至不同的軟硬體，最後會以增強式學習建立一個迷你型自駕車來結束。

幾乎每一章都會從具啟發性的範例開始，透過建構解答的過程來建立問題，並討論解決問題的多種方式，而每一種方式都有不同層次的複雜度與勞力付出。如果您想找的是快速的解決方案，您可能只需要讀某一章的前幾頁就夠了。想要對主題進行深入了解的人就應該讀完整章。當然，基於兩個理由每個人都應該瀏覽每章中的案例探討——它們讀起來很有趣，而且介紹了現實生活中人們如何利用本書所討論的概念來建立真實產品。

我們也討論了深度學習實務人士與產業專家在使用雲端、瀏覽器、行動與邊緣裝置（edge device）來建立真實世界應用時所面臨的實務考量。我們整理了一些實用的「提示與技巧」以及生活故事，來鼓勵讀者建立可以讓某些人過得更好的應用。

給後端 / 前端 / 行動軟體開發人員的話

您應該已經精通程式設計了。即使 Python 對您而言還是一種陌生的語言，我們還是期望您能很快的學會並立即開始使用它。最棒的是，我們沒有期望您已經具有機器學習和人工智慧的背景知識；那就是我們在這裡的原因！我們相信您會從本書下列的焦點中獲益：

- 如何建立直接面對使用者的人工智慧產品？
- 如何快速的訓練模型？
- 如何節省程式碼以及原型設計所需的努力？
- 如何讓模型表現得更好且更節省能源？
- 如何營運與調整規模，以及預估成本？
- 在超過 40 個案例探討以及真實世界範例中發掘如何運用人工智慧。
- 發展深度學習的廣泛知識。
- 發展可應用於新架構（例如 PyTorch）、領域（例如健康照顧、機器人學）、輸入形式（例如視訊、音訊、文字）與任務（例如影像分割、單樣本學習（one-shot learning））的通用技巧。

碁峰 www.gotop.com.tw

給資料科學家的話

您應該已經精通機器學習並可能知道如何訓練深度學習模型。好消息！您可以進一步豐富您的技能，並深化您在這領域的知識以建立真實產品。本書會教您如何進行下列任務，以幫助您的日常以及其他的工作：

- 加速您的訓練，包括在多節點群集（multinode cluster）上的作法
- 建立發展模型和對模型進行除錯的直覺，其中包括超參數調整（hyperparameter tuning），因而可以大幅改善模型的準確度
- 了解您模型的運作原理、揭露資料中的偏見、使用 AutoML 來自動決定最佳的超參數以及模型架構
- 學習其他資料科學家所用的提示與技巧，包括更快的資料蒐集、組織化的追蹤實驗、與他人分享模型，以及獲得您的任務目前最新且最佳的模型資訊
- 使用工具甚至自動的（不用透過 DevOps 團隊）來部署並調整您最佳的模型給真實的使用者

給學生的話

現在是考慮以人工智慧為志業的好時機——它是繼網際網路和智慧型手機之後最重要的技術革命。目前已經有了許多進展，但還有很多尚待開發。我們希望這本書可以成為您在人工智慧職涯的第一步，或者能更進一步的發展更深層的理論知識。最好的一件事是您不用花大錢來購買昂貴的硬體設備。事實上，您可以在瀏覽器上免費的使用威力強大的硬體（感謝您！ Google Colab！）來進行訓練。藉由此書，我們希望您會：

- 藉由開發有趣的專案來開啟人工智慧的職涯
- 藉由學習產業實務來幫助您進行實習與工作機會上的準備
- 藉由建立像是自駕車這樣有趣的應用來發揮您的創造力
- 藉由發揮您的創意來解決人類所面臨的迫切問題，以成為 AI for Good 的優勝者

碁峰 www.gotop.com.tw

給教師的話

我們相信本書所提供的有趣且真實的專案可以成為您課程的補充教材。我們詳細的敘述了深度學習生產線中的每一個步驟，並且說明了如何有效且快速的執行每一步驟的技術。我們在本書中所呈現的每個專案可作為一學期中的個人或群組專案。未來我們會在 *http://PracticalDeepLearning.ai* 上釋出課程的 PowerPoint 投影片。

給機器人學愛好者的話

機器人學很有趣。如果您是機器人學的愛好者，我們應該不用說服您為機器人加上智能會是應該要走的路。日益強大的硬體——例如 Raspberry Pi、NVIDIA Jetson Nano、Google Coral、Intel Movidius、PYNQ-Z2 及其他類似硬體——導引了機器人學的創新。在往工業 4.0 前進的路上，這些平台會愈來愈重要與普及。藉由此書，您可以：

- 學習如何建立與訓練人工智慧模型，以及將它應用在邊緣裝置上
- 評測與比較邊緣裝置之效能、大小、威力、電池，以及成本
- 了解如何選出適用於某一場景之最佳人工智慧演算法以及裝置
- 學習其他自造者建立創新的機器人和機器的方法
- 學習如何在此領域中有更多進展並展示您的作品

本書內容

第 *1* 章，探索人工智慧的景色

我們將對 1950 年代至今日的持續進展進行導覽、分析成為深度學習完美菜單的那些成份、熟悉常見的人工智慧術語與資料集，以及了解人工智慧（responsible AI）的世界。

第 *2* 章，畫中所言為何：用 *Keras* 進行影像分類

我們只需要使用五行 Keras 程式碼就可以深入研究影像分類的世界。然後我們會在將熱圖與視訊重疊並進行預測時，把注意力放在類神經網路上。額外的紅利：我們聽到了來自 Keras 的創造者 François Chollet 激勵人心的故事，了解他對人類所帶來的影響力。

碁峰 www.gotop.com.tw

第 *3* 章，貓狗大戰：用 *Keras* 在 *30* 行內搞定遷移學習

透過遷移學習（transfer learning），我們在新的分類任務上重新使用先前所訓練的網路，進而在極短時間內獲得最好的準確度。接著我們會深入觀察結果，來了解它分類的成效。在此過程中，我們將建立一個會在本書中重複使用的機器學習生產線。額外的紅利：藉由 fast.ai 共同創辦人 Jeremy Howard 的故事，了解數十萬個學生如何使用遷移學習進入人工智慧的領域。

第 *4* 章，建立反向影像搜尋引擎：了解嵌入

如同 Google 反向影像搜尋（Google Reverse Image Search），我們會探討如何使用嵌入（embedding）——影像的內容表達法——以不到十行的程式碼來找出相似的影像，接著會發現當我們探索不同的策略和演算法來加速運算時變有趣了，因為數以千計甚至百萬張的影像能在微秒內得以被搜尋。

第 *5* 章，從菜鳥到大師級預測器：最大化捲積類神經網路的準確度

我們會使用包含 TensorBoard、What-If Tool、tf-explain、TensorFlow Datasets、AutoKeras 及 AutoAugment 等工具來探討可以讓分類器的準確度最大化的策略。其中我們還會以實驗來建立您對人工智慧任務中重要參數的直覺。

第 *6* 章，最大化 *TensorFlow* 之速度與效能：便利的清單

我們藉由一個包含 30 個能夠盡可能提升效率和技巧的清單，將訓練與推論的速度推到極致並最大化您目前硬體的價值。

第 *7* 章，實用工具、提示與技巧

我們將實用技能分散為各種主題與工具，範圍涵蓋安裝、資料蒐集、實驗管理、視覺化及追蹤最先進的研究成果，以建立深度學習的理論基礎。

第 *8* 章，電腦視覺雲端 *API*：*15* 分鐘內開始運行

要聰明的工作，而不是辛勤的工作。我們會在 15 分鐘內學會使用來自 Google、Microsoft、Amazon、IBM 及 ClarifaiWork 的雲端人工智慧平台的強大功能。對於那些無法運用既有 API 來解決的問題，我們也會再使用客製化的分類服務來訓練分類器，並且不需要寫程式。然後我們會讓它們彼此厮殺一番——您會很訝異獲勝者是誰。

碁峰 www.gotop.com.tw

第 *9* 章，使用 *TensorFlow Serving* 與 *KubeFlow* 進行雲端可擴展推論服務

我們將客製化訓練的模型帶到雲端 / 區域電腦上，以服務數百到數百萬規模之需求。還會探討 Flask、Google Cloud ML Engine、TensorFlow Serving 及 KubeFlow，並展示它們的效果、應用情境及性價分析。

第 *10* 章，使用 *TensorFlow.js* 與 *ml5.js* 在瀏覽器上運行人工智慧

每個使用電腦或智慧型手機的人都應該使用過一個軟體程式——瀏覽器。透過基於瀏覽器之深度學習程式庫——包括 TensorFlow.js 和 ml5.js ——來觸及所有的使用者。客座作者 Zaid Alyafeai 陪伴我們經歷像是姿態預測、生成對抗網路（generative adversarial network，GAN）、運用 Pix2Pix 進行影像對影像的翻譯，以及其他更多的技術和任務，但這些並不是在伺服器中而是在瀏覽器中執行。額外的紅利：聽 TensorFlow.js 與 ml5.js 的重要貢獻者談談這些專案的起源。

第 *11* 章，使用 *Core ML* 在 *iOS* 上進行即時物件分類

我們會探討行動裝置上的深度學習，並聚焦於使用 Core ML 的 Apple 生態系。我們在各種的 iPhone 上測試模型效能、探討可以降低應用大小與能源衝擊的策略，並且探討動態模型部署、裝置上訓練，以及如何建立專業應用。

第 *12* 章，使用 *Core ML* 與 *Create ML* 建立 *iOS* 上的 *Not Hotdog*

矽谷的 Not Hotdog 應用程式（來自 HBO）被看作是行動人工智慧界的「Hello World」，所以我們用三種不同的方式建立一個即時版本來向它致敬。

第 *13* 章，食物界的 *Shazam*：使用 *TensorFlow Lite* 和 *ML Kit* 來開發 *Android* 應用程式

我們利用 TensorFlow Lite 結合人工智慧和 Android。接著我們使用 ML Kit（它建立在 TensorFlow Lite 之上）來進行跨平台開發，並用 Fritz 來探討建立自我改善人工智慧應用程式的開發生命週期。在其中我們會檢視模型版本控制、A/B 測試、量測成功率、動態更新、模型優化，以及其他主題。額外的紅利：我們會聽到來自 Pete Warden（行動與嵌入式 TensorFlow 技術長）將人工智慧帶到邊緣裝置的豐富經驗。

第 *14* 章，使用 *TensorFlow* 物件偵測 *API* 建立完美的喵星人定位應用程式

我們會探討四種在影像中找到物件位置的方法。看看這些年來物件偵測的進展，並分析速度與準確度間的取捨。這可以建立像是群眾計數（crowd counting）、臉部偵測及自駕車等案例探討的基礎。

碁峰 www.gotop.com.tw

第 *15* 章，成為自造者：探索邊緣裝置上的嵌入式人工智慧

客座作者 Sam Sterckval 將深度學習導入低耗能裝置上並展示了具有人工智慧能力與不同處理效能的邊緣裝置，包括 Raspberry Pi、NVIDIA Jetson Nano、Google Coral、Intel Movidius 及 PYNQ-Z2 FPGA，開啟了通往機器人學與自造者專案之門。額外的紅利：NVIDIA Jetson Nano 團隊分享人們如何使用他們的開源烹飪書快速的建立具有創意的機器人。

第 *16* 章，使用 *Keras* 進行端到端深度學習模擬自駕車

客座作者 Aditya Sharma 與 Mitchell Spryn 在本章使用了逼近真實的微軟 AirSim 模擬環境，藉由先在此環境中駕駛一遍後再教導人工智慧模型重複此行為，來導引我們訓練一台虛擬汽車。在此過程中，本章涵蓋了數個可應用於自駕車產業的概念。

第 *17* 章，一小時內建造一部自駕車：在 *AWS DeepRacer* 使用增強式學習

由虛擬步入現實，客座作者 Sunil Mallya 展示了如何在一小時內組合、訓練及運行迷你汽車 AWS DeepRacer。再藉由增強式學習的幫助，讓這台車學會自己開車、懲罰犯錯，以及最大化成功率。我們會學到如何應用這個知識在從 Olympics of AI Driving 到 RoboRace（使用全尺寸自駕車）這樣的競賽上。額外的紅利：分享來自 Anima Anandkumar（NVIDIA）及 Chris Anderson（DIY Robocars 的創辦人）對自駕車產業的前景及看法。

本書編排慣例

本書使用以下的字體慣例：

斜體（*Italic*）

指出新字、網址、電子郵件地址、檔名，以及副檔名。中文以楷體表示。

定寬字（`Constant width`）

用於程式列表和在段落中提及的程式元素，例如變數或函數名稱、資料庫、資料型別、環境變數、敘述，以及關鍵字。

定寬粗體字（**`Constant width bold`**）

顯示命令或其他應由使用者輸入的文字。

碁峰 www.gotop.com.tw

第二章

畫中所言為何：用 Keras 進行影像分類

如果您瀏覽了深度學習文獻，可能會偶然發現一連串帶有令人生畏的數學的學術解釋。別擔心！我們會用一個只需要幾行程式碼的影像分類範例，來引導您輕鬆進入深度學習實務。

在本章中，我們會仔細研究 Keras 框架、討論它在深度學習領域中的位置，再透過它利用目前最先進的分類器來分類一些影像。我們會用**熱圖**（*heatmap*）來以視覺化的方式檢視這些分類器的運作成效。透過這些熱圖，我們會製作一個有趣的專案以分類視訊中的物件。

在第 14 頁的「完美深度學習解決方案的配方」中曾提到，我們需要四個材料來建立我們的深度學習配方：硬體、資料集、框架及模型。讓我們看看它們在本章中的角色：

- 我們從最簡單的開始：**硬體**。即便是便宜的筆記型電腦也足以執行本章要做的事。您也可以開啟一個 GitHub 筆記本（參見 *http://PracticalDeepLearning.ai*）來在 Colab 中執行本章的程式碼。這只需要點擊幾次滑鼠即可。
- 因為我們還不想要訓練類神經網路，所以目前還不需要**資料集**（除了幾張用來測試的樣本照片）。
- 接著是**框架**。本章的標題中有 Keras 這個字，所以那就是我們現在會用到的。事實上，我們在本書中只要有訓練需求的話，大多會使用 Keras。

碁峰 www.gotop.com.tw

- 接近深度學習問題的一種方式是取得一個資料集、寫程式訓練它、花上一大堆時間和能源（人類和電力都是）來訓練這個模型，再使用它進行預測。不過我們不是懲罰魔人。因此，我們會使用**預先訓練好的模型**。畢竟研究社群已經花了無數心血來訓練且發表許多已經公開可用的標準模型了。我們會重新使用一種名為 ResNet-50 的知名模型，它是 2015 年的 ILSVRC 獲勝者 ResNet-152 的兄弟。

本章中您會取得一些包含程式碼的實作練習。我們都知道，最好的學習方式就是動手做。雖然您可能會覺得奇怪，這背後的理論是什麼？我們將會在後面的章節中說明，並以本章作為基礎深入發掘 CNN 的基本要領。

Keras 出場

就如第 1 章所討論的，2015 年 Keras 開始時是作為在其他程式庫之上易於使用的抽象層，它讓快速原型設計變為可能。這會使深度學習的初學者比較容易學習。在此同時，它也藉由幫助深度學習專家快速的重複實驗來提高生產力。事實上，*Kaggle.com*（主辦資料科學競賽）主要的獲勝團隊多數都使用 Keras。終於在 2017 年時 TensorFlow 直接包含了完整的 Keras 實作，因而結合了高可調性、效能，以及 TensorFlow 的巨大生態系與 Keras 的方便性。在網路上，我們經常看到把 TensorFlow 的 Keras 稱之為 `tf.keras`。

在本章與第 3 章中，我們只用 Keras 來寫程式。這包含模組化函數（boilerplate function），像是檔案讀取、影像操作（擴增）等等。我們這麼做主要是為了學習方便。從第 5 章以後，我們會漸漸開始直接使用更多的原生高效能 TensorFlow 函數，以獲得更多的可設定性以及控制。

創造者的話

François Chollet，Keras 的創造者、人工智慧研究者，以及 *Deep Learning with Python* 的作者

我本來是為了要自己用才開始接觸 Keras 的。當時是 2014 年末和 2015 年初，尚未具有紮實可用性並強烈支援 RNN 以及 convnet 的深度學習框架可以選擇。那時我並沒有刻意民主化深度學習，只是建立自己想要的工具而已。不過隨著時間過去，我看到有很多人透過 Keras 來接觸深度學習，還用它來解決一些我從來不知道的問題。對我來說，這實在是太美妙了。它讓我理解到深度學習可以用顛覆性的方式部署到許多機器學習研究者意想不到的領域。有這麼多人可以經由使用這些技術而在他們的工作中受益。因此，我開始關心如何盡可能讓

碁峰 www.gotop.com.tw

更多人可以懂得深度學習。那是讓人工智慧可以完全發揮潛力的唯一方式——讓它到處可用。如今在 TensorFlow 2.0 中，Keras API 加強了深度學習的威力，憑藉的是具有生產力與有趣的工作流程、適用從研究者到應用程式的不同特性使用者，也包含部署。我很期待看到您使用它來建立東西！

預測影像的類別

以外行的話來說，影像分類回答了這個問題：「這張影像包含什麼物件？」更精確的說，「這張影像有多少機率包含 *X* 物件？」其中 *X* 來自於事先定義的物件類別串列。如果此機率高於一個最小閾值（threshold），那麼這個影像就很有可能包含一或多個 *X* 實例。

一個簡單的影像分類生產線會包含下列步驟：

1. 載入一張影像。
2. 將它的大小調整成事先決定的大小，例如 224×224 像素。
3. 將像素值縮放到範圍 [0,1] 或 [–1,1]，也被叫作正規化（normalization）。
4. 選擇一個預先訓練好的模型。
5. 在影像上執行此預先訓練好的模型，以獲得一串列的類別預測以及其對應機率。
6. 顯示一些具有最高機率的類別。

GitHub 連結放在網站 *http://PracticalDeepLearning.ai* 上。在 `code/chapter-2` 您就能找到包含所有步驟細節的 Jupyter 筆記本 `1-predict-class.ipynb`。

我們從 Keras 以及 Python 套件匯入所有必要的模組開始：

```
import tensorflow as tf
from tf.keras.applications.resnet50 import preprocess_input, decode_predictions
from tf.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
```


碁峰 www.gotop.com.tw

接下來，我們載入並顯示想要分類的影像（參見圖 2-1）：

```
img_path = "../../sample-images/cat.jpg"
img = image.load_img(img_path, target_size=(224, 224))
plt.imshow(img)
plt.show()
```

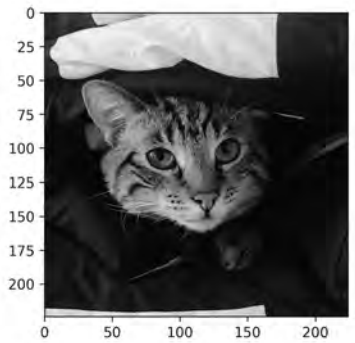

圖 2-1　顯示輸入檔的圖表

沒錯，它是一隻貓（雖然檔名已經透露這件事）。那正是我們的模型理想的預測結果。

影像的簡單複習

在我們深入介紹影像是如何處理之前，最好先來了解一下影像是如何儲存資訊的。在最基本的層次上，影像是排列在矩形格柵的像素集合。根據影像類型的不同，每個像素可以包含 1 到 4 個部分（也稱為**成份**（*component*）或**頻道**（*channel*））。對我們要用的影像而言，這些成份代表紅、綠、藍（RGB）三種顏色的亮度。它們一般的長度都是 8 位元，所以值會落在 0 到 255（也就是 $2^8 - 1$）之間。

在將影像餵給 Keras 前，我們想先將它轉換為標準格式。這是因為那個預先訓練好的模型所期待的是具有特定大小的輸入。而在我們的案例中，標準化指的是將影像調整成 224×224 像素的大小。

大多數的深度學習演算法會期望輸入一堆影像。不過，當我們只有一張影像時該怎麼辦呢？當然，我們要建立一批次（batch）的影像！實際上就是建立包含同一個物件的影像陣列。另一種處理方式是將維度的數量由 3（代表影像的 3 個頻道（channel））擴展到 4（多出來的一個是陣列的長度）。

碁峰 www.gotop.com.tw

如果這樣還是不夠清楚，請看看以下情境：對於一批次包含 64 張大小為 224×224 像素的影像，每個像素都包含 3 個頻道（RGB），代表這批次影像的物件的形狀會是 64×224×224×3。在下面的程式碼中，我們只用了一張 224×224×3 的影像，我們會藉由將維度從 3 擴展到 4 來建立一個只包含那張影像的批次。這個新建立的批次形狀會是 1×224×224×3：

```
img_array = image.img_to_array(img)
img_batch = np.expand_dims(img_array, axis=0) # 增加維度數量
```

在機器學習中，當餵給模型的資料是落在一致的範圍內時，模型的表現會最好。範圍一般是包含 [0,1] 和 [–1,1]。如果影像的像素值是在 0 到 255 之間，在輸入影像上執行 Keras 的 `preprocess_input` 函數會將每一個像素正規化到一個標準範圍內。**正規化**（*normalization*）或是**特徵縮放**（*feature scaling*）是影像前置處理的核心步驟，使影像可以適於深度學習。

現在模型進場了。我們會使用一個稱為 ResNet-50 的**捲積類神經網路**（*Convolutional Neural Network*，CNN）。我們應該要問的第一個問題是：「我要到哪裡去找到那個模型？」當然，我們可以在網際網路上搜尋以找到某個和我的深度學習框架（Keras）相容的東西。**不過不是所有人都有空做這件事！**幸運的是，Keras 喜歡把事情簡單化，將它以一個函數呼叫的方式提供給我們使用。在我們第一次呼叫這個函數後，會從一個遠端伺服器下載那個模型並儲存於區域儲存空間：

```
model = tf.keras.applications.resnet50.ResNet50()
```

用這個模型來進行預測時，其結果會包含每個類別的預測機率。Keras 也提供了 `decode_predictions` 函數，它會告訴我們影像中所包含的每個物件的機率。

現在讓我們來看一個好用函數的整段程式碼：

```
def classify(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    model = tf.keras.applications.resnet50.ResNet50()
    img_array = image.img_to_array(img)
    img_batch = np.expand_dims(img_array, axis=0)
    img_preprocessed = preprocess_input(img_batch)
    prediction = model.predict(img_preprocessed)
    print(decode_predictions(prediction, top=3)[0])

classify("../../sample-images/cat.jpg")

[('n02123045', 'tabby', 0.50009364),
 ('n02124075', 'Egyptian_cat', 0.21690978),
 ('n02123159', 'tiger_cat', 0.2061722)]
```


碁峰 www.gotop.com.tw

這張影像的預測類別是不同類型的貓科動物。為什麼它不會只簡單的預測為一個字詞「貓」呢？簡單的說，是因為 ResNet-50 模型是以具有細緻類別的資料集訓練的，其中並沒有包含更廣義的「貓」。稍後我們會更詳細的調查這個資料集，現在先讓我們載入另一張樣本影像（參見圖 2-2）：

```
img_path = '../../sample-images/dog.jpg'
img = image.load_img(img_path, target_size=(224, 224))
plt.imshow(img)
plt.show()
```

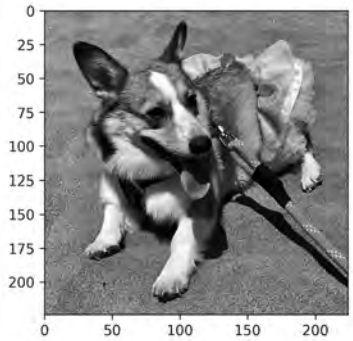

圖 2-2　顯示檔案 dog.jpg 內容的圖表

我們再一次執行剛才那個好用的函數：

```
classify("../../sample-images/dog.jpg")

[(u'n02113186', u'Cardigan', 0.809839),
 (u'n02113023', u'Pembroke', 0.17665945),
 (u'n02110806', u'basenji', 0.0042166105)]
```

就我們所預期的，我們會得到不同的犬科（而不是「狗」類別）的結果。如果您對科基（Corgi）犬不熟悉，「corgi」這個字其字面上的意思就是威爾斯語的「矮人狗」。卡提根（Cardigan）和潘布魯克（Pembroke）是柯基犬的子品種，所以彼此間看來很相似。難怪我們的模型也是這麼認為。

請注意每一個類別的預測機率。通常具有最高機率的預測會被視為答案。另一種選擇是將所有機率超過某個事先定義的閾值的類別也視為答案。在狗範例中，如果我們將閾值設為 0.5，卡提根柯基犬就會是我們的答案。

碁峰 www.gotop.com.tw

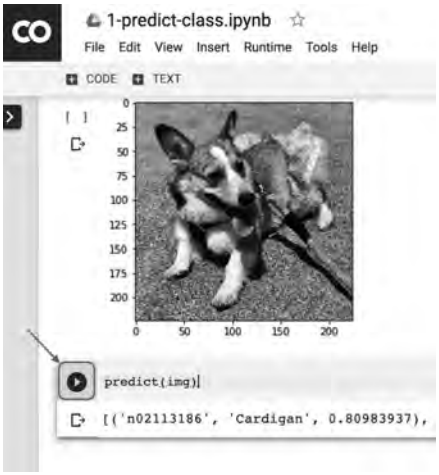

圖 2-3　使用瀏覽器在 Google Colab 上執行筆記本

您不用安裝 Google Colab 就可以在瀏覽器中執行本章的程式碼。只要找到您想實驗的每個 GitHub 筆記本頂端的「Run on Colab」連結，然後點擊「Run Cell」按鈕，就會執行那個儲存格中的程式碼，如圖 2-3 所示。

模型的深入調查

我們從模型獲得預測結果了，太棒了！不過是哪些因子導致那些預測的？這裡我們有幾個問題必須先了解：

- 模型是用什麼資料集訓練的？
- 還有其他模型可以用嗎？它們有多好？我可以在哪裡找到它們？
- 我的模型為什麼會預測出那個結果？

在本節中我們會回答這些問題。

碁峰 www.gotop.com.tw

ImageNet 資料集

讓我們檢視一下用來訓練 ResNet-50 的 ImageNet 資料集。顧名思義，ImageNet（*http://www.image-net.org/*）是由影像所構成的網路；也就是說，由組織成網路的影像所構成的資料集，如圖 2-4 所展示的。它是以階層式排列（就像是 WordNet 階層），其中父節點涵蓋了所有在此階層中可能的影像集合。例如，在「動物」這個父節點中就包含魚、鳥、哺乳類動物、無脊椎動物等等。每個類別會有多個子類別，而它們又有子類別，依此類推。例如，類別「美國水獵犬（American water spaniel）」位於根節點之下的第八層。狗的類別包含了分布在五個階層的 189 個子類別。

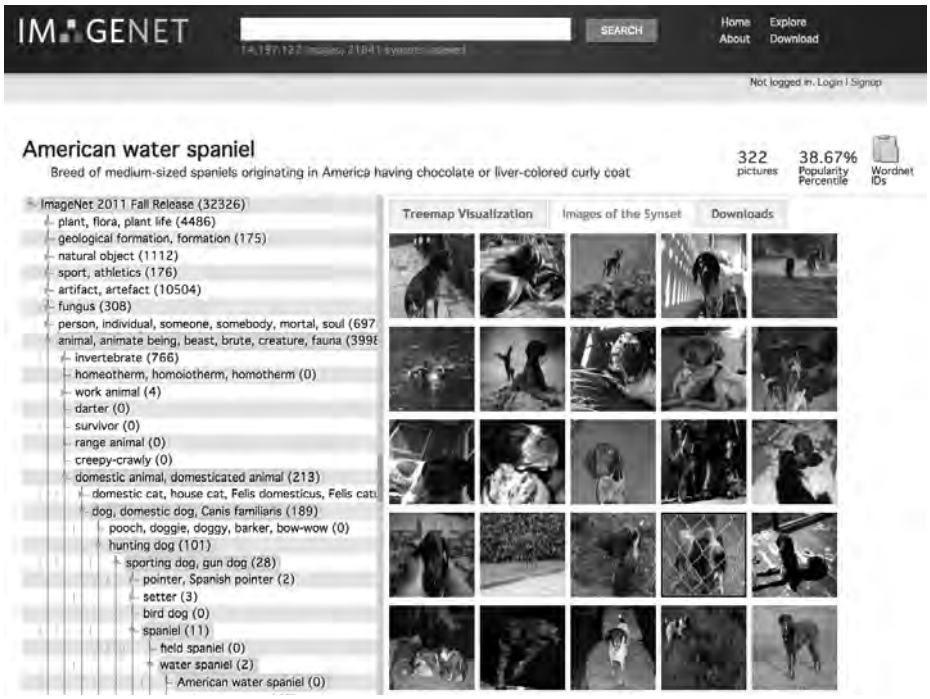

圖 2-4　ImageNet 資料集的類別和子類別

在視覺上，我們發展了圖 2-5 中的樹狀圖來幫您了解 ImageNet 資料集所包含的各式各樣高層實體。這個樹狀圖（treemap）也呈現了建立 ImageNet 資料集其不同類別的相對百分比。

碁峰 www.gotop.com.tw

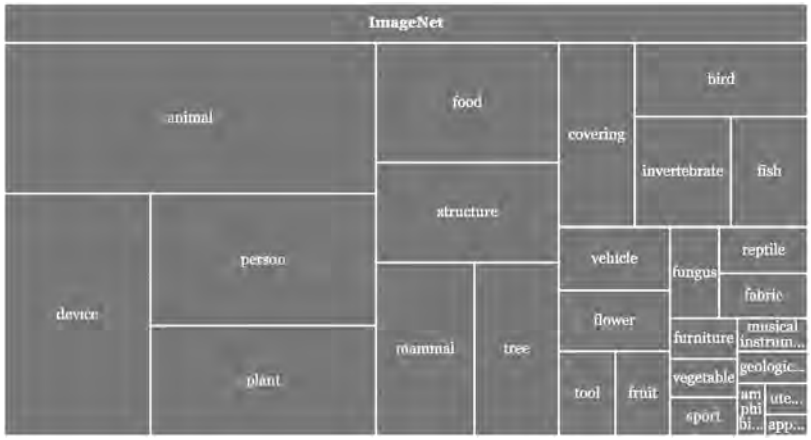

圖 2-5　ImageNet 及其類別的樹狀圖

ImageNet 資料集是知名的 ILSVRC 的基礎，它開始於 2010 年，用以評測電腦視覺的進展以及要求研究者在物件分類這類的任務上進行創新。回想第 1 章中曾提到，ImageNet 挑戰每年的參賽作品和準確度都大幅增加。一開始時，錯誤率將近 30%，現在則是 2.2%，已經比一般的人類在做同一件事時表現得更好。這個資料集以及挑戰被認為是電腦視覺最近進展的一個最大原因。

等一下，人工智慧表現得比人類還好？如果資料集是由人類所創造的，那麼人類的準確度不應該是 100% 嗎？好吧，資料集是由專家所創建的，每一張影像又由好幾個人驗證過。然後，史丹福大學的研究者（現在是特斯拉的人）Andrej Karpathy 企圖要指出一般人在 ImageNet-1000 的表現如何。結果發現他的準確度是 94.9%，比我們所預期的 100% 少。Andrej 花了一整個星期痛苦的看了超過 1,500 張影像，每張影像用了大約一分鐘來標註它。為什麼他會將 5.1% 的影像分類錯誤呢？答案有點微妙：

細緻辨識

對許多人來說，要分辨西伯利亞哈士奇犬和阿拉斯加愛斯基摩犬真的很困難。十分熟悉狗的品種的人，或許可以根據更細微的細節來分辨它們。但結果是，類神經網路比人類還容易學到這些更細微的細節。

類別未感知

不是所有的人都可以感知到 120 種狗，更不用說那 1,000 種類別中的每一個了。不過人工智慧可以，因為它是用那資料集訓練的。

碁峰 www.gotop.com.tw

和 ImageNet 一樣，像 Switchboard 這樣的語音資料集報告的語音轉譯錯誤率為 5.1%（恰好與 ImageNet 相同）。很明顯地，人類是有極限的，而人工智慧正逐漸擊敗我們。

另一個造成這種快速改善的關鍵原因是，研究者公開的分享了他們使用像是 ImageNet 這樣的資料集所訓練的模型。在下一節中，我們會更詳細的介紹模型重利用。

模型動物園

模型動物園（model zoo）是一個讓組織或個人可公開上傳他們所訓練的模型的地方，供其他人可以重利用並加以改善。這些模型可以用任何框架訓練（例如 Keras、TensorFlow、MXNet）、用在任何任務（分類、偵測等），或用任何資料集訓練（例如 ImageNet、Street View House Numbers（SVHN））。

模型動物園的傳統始於 Caffe，它是由加州大學柏克萊分校所發展最早的深度學習框架之一。用一個具有好幾百萬張影像的資料庫從無到有的訓練深度學習模型，將會需要好幾個星期的訓練時間以及大量的 GPU 計算能源，所以是件困難的任務。研究社群認知到這會是一個瓶頸，所以參與 ImageNet 競賽的組織們在 Caffe 的網站上開放了他們訓練模型的原始碼。其他的框架很快也跟進了。

當開始一個新的深度學習專案時，先探索一下是否有類似的模型已經用類似的資料庫訓練並用來執行類似的工作，會是一個好主意。

Keras 的模型動物園（*https://keras.io/applications/*）集合了使用 Keras 框架在 ImageNet 資料集上訓練完成的不同架構。請見表 2-1。

表 2-1　預先訓練之 ImageNet 模型的架構細節

模型	大小	前一準確度	前五準確度	參數	深度
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.9	143,667,240	26
ResNet-50	98 MB	0.749	0.921	25,636,712	50
ResNet-101	171 MB	0.764	0.928	44,707,176	101
ResNet-152	232 MB	0.766	0.931	60,419,944	152
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
NASNetMobile	23 MB	0.744	0.919	5,326,716	—
NASNetLarge	343 MB	0.825	0.96	88,949,818	—
MobileNet	16 MB	0.704	0.895	4,253,864	88

碁峰 www.gotop.com.tw

模型	大小	前一準確度	前五準確度	參數	深度
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

「前一準確度（top-1 accuracy）」那欄指出最好的預測結果就是正確答案的比例，「前五準確度」則是前五名中至少有一個是正確的比例。類神經網路的「深度」是指網路中包含多少層。「參數」欄則是指模型的大小；也就是說，模型中具有多少個權重：參數愈多，模型愈「重」，它預測速度也愈慢。在本書中我們經常使用 ResNet-50（因為高準確度而成為研究論文中最常被引用的架構）以及 MobileNet（在速度、大小與準確度上達到不錯的平衡）。

類別激發圖

在 UX 研究中知名的影像顯著性（saliency）會試著回答以下問題：「影像中的哪個部分是使用者會關注的？」眼動追蹤（eye-tracking）的研究以及熱圖表達法更突顯了這個問題。例如，大粗體字型或者人臉通常會比背景獲得更多的關注。我們很容易就可以猜到這些熱圖對設計師和廣告主會是很有用的，因為他們可以調整他們的內容來最大化使用者的關注。受到這個人類版的顯著性的啟發，如果能知道類神經網路正在關注影像的哪個部分不是很棒嗎？這正是接下來我們的實驗所要做的。

在我們的實驗中，我們會將一張**類別激發圖**（*class activation map*）（或者比較口語的說法叫熱圖）重疊在一段視訊上，以了解網路在關注什麼。熱圖告訴我們像是「在這張照片中，這些像素負責偵測 dog 類別」這樣的事情，其中「dog」是具有最高機率的類別。「熱的」像素會用較溫暖的顏色，像是紅色、橘色和黃色來代表，而「冷的」像素會用藍色來代表。像素愈「熱」，它提供給預測的信號愈強。圖 2-6 讓我們有更清楚的概念。（您可以參考本書的 GitHub 來取得原始彩色影像。）

圖 2-6　狗的原始影像以及所產生的熱圖

碁峰 www.gotop.com.tw

在本書的 GitHub（參見 *http://PracticalDeepLearning.ai*）中，請您移至 *code/chapter-2* 資料夾，會找到一個方便的 Jupyter 筆記本 *2-class-activation-map-on-video.ipynb*，其中描述了下列步驟：

首先，我們需要用 `pip` 安裝 `keras-vis`：

```
$ pip install keras-vis --user
```

然後，在單一影像上執行視覺化腳本來產生它的熱圖：

```
$ python visualization.py --process image --path ../sample-images/dog.jpg
```

我們應該會看到一個名為 *dog-output.jpg* 的新增檔案，它會並列顯示原始影像和它的熱圖。如同在圖 2-6 所看到的，右半部的影像指出「熱區」以及正確的預測「卡提根」（也就是威爾斯柯基犬）。

接下來，我們想要視覺化視訊中畫面的熱圖。因此，我們需要開放原始碼的多媒框架 `FFmpeg`。您可以從 *https://www.ffmpeg.org* 下載符合您作業系統的版本以及安裝說明。

我們會使用 `ffmpeg` 來將一段視訊切成個別圖框（每秒 25 張圖框），然後再對每張圖框執行視覺化腳本。我們必須先建立一個目錄來儲存這些圖框，並將目錄名稱傳給 `ffmpeg` 作為命令的一部分：

```
$ mkdir kitchen
$ ffmpeg -i video/kitchen-input.mov -vf fps=25 kitchen/thumb%04d.jpg -hide_banner
```

接著，我們用上一步驟中包含畫面的目錄路徑來執行視覺化腳本：

```
$ python visualization.py --process video --path kitchen/
```

我們應該會看到新建立的 *kitchen-output* 目錄中包含了所有來自輸入目錄的畫面的熱圖。

最後，使用 `ffmpeg` 將畫面編成一段視訊：

```
$ ffmpeg -framerate 25 -i kitchen-output/result-%04d.jpg kitchen-output.mp4
```

太完美了！結果是原始影像以及它的熱圖疊在它複本上的並列畫面。這是一個很有用的工具，特別是用來發掘模型是否有學到正確的特徵，或者它是否在訓練過程中撿到流浪寶器。

想像一下，產生熱圖來分析我們的訓練模型或某個預先訓練模型的優缺點。

碁峰 www.gotop.com.tw

您應該用您自己的智慧型手機拍一段視訊並在此檔案上執行前述的腳本。別忘了將您的視訊發表在 Twitter 上並標註 @PracticalDLBook（*https://www.twitter.com/PracticalDLBook*）！

熱圖是一種用視覺方式偵測資料中偏見的好方法。模型預測的品質十分仰賴用來訓練它的資料。如果資料是有偏見的，那將會影響預測。一個好的範例（雖然可能只是都市傳說）是美國陸軍想要使用類神經網路來偵測偽裝在樹林的敵方坦克[1]。想要建立模型的研究者拍了照片—— 50% 包含了偽裝坦克，另外 50% 只有樹。模型訓練後的準確度是 100%。這樣就可以慶祝了嗎？不，遺憾的是，當美國陸軍測試時並不是那麼一回事。此模型的表現極差——和亂猜沒兩樣。調查後發現，有坦克的照片是在陰天拍的，而沒有坦克的照片是在晴天拍的。類神經網路所找的是天空而不是坦克。如果那些研究者有用熱圖來視覺化模型的話，他們可以在很早的時候就抓到問題。

當我們蒐集資料時，必須從一開始就要對那些會污染我們模型學習的潛藏偏見提高警覺。例如，當為了建立食物分類器而蒐集影像時，我們要確認其他的物品，例如盤子和餐具，不會被當作食物來學習。否則，出現在其中的筷子可能會被分類成炒麵。另一個用來定義這件事的詞是**共現**（*co-occurrence*）。食物經常和餐具共同出現，所以要小心這些器具溜進您分類器的訓練過程中。

總結

本章中，我們使用 Keras 簡單介紹了深度學習世界。它是一種容易使用卻威力強大的框架，在後面幾章中我們還會用到。我們觀察到，其實不需要蒐集數百萬張影像和使用威力強大的 GPU 來訓練客製化的模型，因為我們可以使用預先訓練的模型來預測影像的類別。深入研究像是 ImageNet 這樣的資料集後，我們了解這些預先訓練的模型可以預測出什麼樣的類別。我們也學到可以從模型動物園中找到這些模型。

在第 3 章中，將會探討如何調整既有的預先訓練模型來對它原先沒想到的輸入類別進行預測。而在本章，我們的做法是在不需數百萬張影像以及大量硬體資源上來得到輸出。

1 「Artificial Intelligence as a Positive and Negative Factor in Global Risk」（*https://oreil.ly/-svD0*） by Eliezer Yudkowsky in *Global Catastrophic Risks* (Oxford University Press).

碁峰 www.gotop.com.tw