前言

AI 和機器學習都是可以改變世界的革命性科技，不過僅限於有開發者可以使用不錯的 API 來取得這些科技所帶來的進展才行。

其中一項進展是能夠在瀏覽器中執行機器學習模型，使得應用可以表現得更聰明。

TensorFlow.js 的興起告訴我 AI 時代已經來臨。不再是只適用於使用超級電腦的資料科學家；它現在也可以讓每天以 JavaScript 寫作的數以百萬計的開發者用了。不過還是有道鴻溝存在。建立模型的工具和技術還是掌握在那些瞭解 Python、NumPy、繪圖處理器（GPUs）、資料科學、特徵定模、監督式學習、張量、還有眾多拗口且你可能還不認識的詞彙的人手上！

Gant 在這本書中做到的是開門見山的教導你，以網頁開發者角色使用 JavaScript 和瀏覽器的情況下，所需知道的重要事情。他會介紹 AI 和機器學習的概念，並且聚焦於如何在你喜愛的平台上使用它們。

我經常聽到想要使用機器學習的開發者問到，「我在哪裡可以找到可以再利用的東西？我不想學著成為機器學習工程師只是為了弄清楚這些東西是否適合我！」

Gant 以本書回答了那個問題。你可以在 TensorFlow Hub 中找到所想要使用的預製模型。你也會學到如何站在巨人的肩膀上，學習怎麼來選擇性的使用那些以數百萬筆資料花了數千小時訓練的模型的一部份，並瞭解如何將它們的學習轉移到你自己的模型中。然後你只要把它丟到網頁中讓 JavaScript 去完成其他的事！

有開發人員問，「我如何不用再花大量時間重新訓練就可以在我在乎的平台上使用機器學習？」

本書深入的說明這點——告訴你如何橋接 JavaScript 以及使用 TensorFlow 所訓練的模型間的鴻溝。從基本變數和張量間的資料轉換，到將輸出機率剖析成文字，本書會一步步指引你如何和你的網站緊密的整合。

有開發人員問我，「我想超越其他人的工作以及簡單的原型。作為 web 開發人員，我可以做得到嗎？」

我還是要說，可以。當你讀完這本書時，你不但會熟悉怎麼使用模型，Gant 也會告訴你建立自己的模型的所有細節。你會學到如何訓練一個可以辨識影像內容的卷積神經網路這樣的複雜模型，而且全部都用 JavaScript 來完成。

2020 年 10 月的調查結果顯示全世界有 1240 萬名 JavaScript 開發者。其他的調查也顯示全球有大約 30 萬名 AI 實務工作者。擁有了 TensorFlow.js 技術以及本書所提到的技能後，你——親愛的 JavaScript 開發人員，就可以加入 AI 世界。本書是一個很棒的起點。

享受你的旅程吧！

— *Laurence Moroney*
2021 年 3 月

序

「如果你選擇不做決定，你仍然已經做出了選擇。」

— Geddy Lee（Rush）

我們開工吧

我們總是事後諸葛亮。「我應該在比特幣還是 X 元時買的」或者「我應該在新創公司 Y 變有名前應徵工作的。」世界充斥著決定我們會更好或更差的時刻。時間從不倒退，但我們以前所做的選擇會隨著我們而前進。你很幸運擁有此書並可以在此刻做出決定。

軟體產業的基礎正隨著人工智慧而改變。決定會如何改變的正是那些抓住並形塑未來世界的人。機器學習是建立新的可能性的冒險，當它和廣被使用的 JavaScript 聯合後，其間的界限就逐漸消失了。

就像我在有關 AI 的演講中說的，「如果你建立的軟體只能帶你走到這裡，那你就不會在這裡。」所以讓我們開始吧，看看我們的想像力可以把我們帶向何方。

為何要用 TensorFlow.js？

TensorFlow 是市面上最受歡迎的機器學習框架之一。它是由 Google 中的聰明腦袋們所支援，而且被全世界許多具有影響力的公司所採用。TensorFlow.js 是 TensorFlow 的 JavaScript 框架，而且它優於所有競爭對手。簡單說，如果你想要 JavaScript 框架所具有的威力，只有一種選擇可以做得到。

誰該讀本書？

兩個主要族群會享受本書的內容並且受益：

JavaScript 開發人員

如果你熟悉 JavaScript，不過以前從未接觸過機器學習，本書會是你的嚮導。它以框架為主，讓你可以活躍在既務實又令人興奮的創作中。你會透過建立各種專案的經驗來理解機器學習的基礎。我們不會逃避數學或者深入的概念，但也不會把它弄得過於複雜。如果你想要用 JavaScript 建立網站又想擁有超能力的話，就讀本書吧。

AI 專家

如果你熟悉 TensorFlow 或甚至線性代數（linear algebra）的基礎原理，本書會用無數的範例來教你如何將你的技能帶入 JavaScript。在此你會看到各種核心概念是如何以 TensorFlow.js 框架來展示與呈現的。這讓你可以把你的知識應用在四處可見的邊緣裝置（edge device）上，像是瀏覽器或物聯網（IoT）。閱讀本書來學習如何將你的創作帶入無數的交談式裝置中。

本書需要適度的閱讀與理解 JavaScript 的能力。

本書概觀

在草擬此書架構時，我明白了我必須做出選擇。要麼創造一個使用小型又易理解的範例來建立各種深度學習應用的冒險歷程，要麼選擇一個單一路徑來講述這些概念的不斷發展的故事。在我徵詢過朋友和追隨者的意見後，顯然後者才是需要的。為了讓本書有合理的頁數，我選擇捨棄所有 JavaScript 框架，並專注於 AI 視覺方面的實務過程。

每一章的最後都有一些問題和挑戰來測試你的理解程度。本章挑戰小節的設計是用來將課程融入你的 TensorFlow.js 肌肉記憶中。

章節介紹

第 1 章和第 2 章以核心概念和具體範例開始。這種陰陽互補作法反映了本書的教學風格。每一章都建構在前面章節所提及的課程、詞彙、以及函數上。

第 3 到 7 章則要讓你具有瞭解和實作既有 AI 工具與資料的視野。你將可以建立亮眼的程式庫，或將模型部署在由大量資料科學家建立的專案中。

第 8 到 11 章開始讓你瞭解 TensorFlow.js 的威力。你將可以用 JavaScript 來訓練模型，我深信這是全書中最有趣且令人興奮的部份。

第 12 章是最後的挑戰。最後一章展現了一個總結專案，讓你能夠使用本書所提供的一切知識，並使用你自己的方式完成它。

重點整理

讀完本書後，不論你之前是否有經驗，你都可以用 TensorFlow.js 來尋找、實作、調整、和建立機器學習模型。你將有能力可以識別出網站中的機器學習應用，而後繼續把它實作完成。

本書編排慣例

本書使用以下的字體慣例：

斜體字（*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、以及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。

定寬斜體字（*`Constant width italic`*）

顯示應該被使用者所提供或由語境（context）決定的值所取代的文字。

此圖示表示提示或建議。

第一章

AI 是魔法

「任何夠先進的技術和魔法沒有差別。」

—Arthur C. Clarke

好吧，AI 並不是真的魔法。事實上 AI 比目前的技術更進一步，讓它感覺像魔法。要解釋一個有效的排序演算法很容易，但是深入智能（intelligence）這塊則頗具爭議性，並且將我們所有人帶入一個全新的技術力層次。TensorFlow.js 的智慧和能力使這種爆炸性的威力提升成為可能。

半個多世紀以來，科學家和工程師們重新創造了 AI 中的隱喻性的輪子，並微調了控制它的機制。當我們深入探討本書中的 AI 時，我們將以具彈性卻又強固的 TensorFlow.js 框架來理解那些概念，並且用它在不斷擴展的 JavaScript 領域中實現我們的想法。沒錯，就是 JavaScript，全世界最受歡迎的程式語言 [1]。

此處的概念和定義讓你具有電光眼。你將能解析縮寫和流行語來查看和理解無處不在的 AI 基礎架構。AI 和機器學習概念會變得清晰，而本章中的定義將成為識別核心原則的參考，這些原則將推動我們的學術發展進入 TensorFlow.js 世界。

我們會：

- 釐清 AI 與智能領域的差別
- 討論機器學習的類型
- 回顧並定義常用術語

1 Programming language stats: *https://octoverse.github.com*

- 透過 TensorFlow.js 交叉檢視概念

我們開始吧！

如果你已經熟悉 TensorFlow.js 以及機器學習的術語、哲學、還有基本應用的話，你可以直接跳到第 2 章。

JavaScript 中的 AI 之路

TensorFlow.js，用最通俗的定義來說，就是在 JavaScript 中用來處理特定 AI 概念的一種框架。就這樣。你很幸運，你在對的時間讀了對的書。AI 產業革命才剛開始。

電腦問世後，擁有電腦的人可以完成一些不可能的任務。他們可以破譯密碼、從大量資料中瞬間獲取資訊、甚至像和人類交手一樣玩遊戲。以前人們不可能做到的事不但成為可能，甚至還變成日常之事。從這個關鍵的數位發明出現開始，我們的目標只有想讓電腦可以「做得更多」。身為人類，我們可以做任何事情，但不是所有事。電腦給了我們新的能力，擴展了我們的極限。許多人一生都在磨練一些技能，但能成為專長的很少。我們一生中都在建立有價值的成就，有一些人在某件事上成為世界上最好的，而那項技能只能靠運氣、資訊和無數日子的努力才能獲得……直到現在。

AI 讓我們直接進入隊伍的前列；勇敢的建造以往從未建造過的事物。每一天我們都可以看到企業和學者一再的獲得計算上的進展。我們正站在新產業的入口，邀請我們成為改變世界的一份子。

你坐在駕駛座上，朝著下個重大目標前進，而本書正是你的方向盤。我們學習 AI 魔法的旅程只會受限於你的想像力。配備了 JavaScript 支援的機器學習後，你將可以善用攝影機、麥克風、即時更新訊息、位置和其他實體感測器、服務和裝置！

我知道你會問，「那為什麼 AI 以前不行？為什麼現在才重要？」要理解此事，你必須知道人類尋找再生智慧（reproduced intelligence）的過程。

何謂智慧？

要介紹邁向機器智慧的想法概念以及途徑都可以寫成一堆書了。而且就如同所有哲學的結論一樣，每種關於智慧的具體敘述都飽受爭論。你不需要知道一切，不過我們需要瞭解 AI 領域，以讓我們可以瞭解這本有關 TensorFlow.js 的書應該著重些什麼。

幾世紀以來詩人和數學家們認為人類思想只不過是既有概念的組合。生命的出現被認為是神的設計；我們只是從元素「製造」而來。希臘神話故事中有一位發明之神赫菲斯托斯（Hephaestus）創造了可以行走的自動青銅機器人當作士兵使用。基本上這些就是第一批的機器人。機器人和智慧的概念以來自這個古老傳說的一種終極和神聖工藝而深植人心。自動化的巨人戰士塔羅斯（Talos）（*https://oreil.ly/L8Ng2*）因被設計成保護克里特島而聞名。雖然青銅機器人並不存在，這故事仍然引燃了對機械的渴望。數百年來，動作古物（animatronic antiquity）經常被認為通往具備人類「智慧」的途徑。幾世紀後，我們開始看到生命在模仿藝術。當我還小時，記得有一次去美國的 Chuck E. Cheese 餐廳，裡面有給小孩子看的動態機器人音樂表演。我記得自己有那麼一瞬間相信那個每天由傀儡操作的電子音樂會是真的。我這樣的啟發來自於和科學家追求智慧一樣的火花。這個火花一直存在著，透過故事、娛樂、以及現在的科學流傳。

幾世紀來，機器可以自動又聰明的進行工作這樣的概念持續進展，我們致力於定義這些概念性的實體。學者在論文中持續研究推論和學習，同時將他們的術語保持在「機器」和「機器人」領域。機械智慧的模仿總是因為缺乏速度和電力而受阻。

幾百年來我們都認為智慧的概念只存於人心而與機器結構無關，直到終極機器——電腦——出現。計算就像多數機器一樣，整台裝置只有一個目的。由於計算的進展，出現了一個新術語來說明不斷進步的智慧，這在很大程度上反映了人類的智力。AI 這個詞彙代表人工智慧（artificial intelligence），它是在 1950 年代出現的 [2]。由於電腦演變成多用途的，哲學和學門開始合而為一，模仿智慧的概念從神話變成科學研究領域，每個電子量測裝置都變成電腦的新感官，也是電子和智慧科學激動人心的機會。

簡單來說，我們會有可以和人類互動以及模仿人類動作的電腦。對於人類行為的模仿讓我們想要稱此為「智慧」。**人工智慧**是這些戰略行動的總稱，而無論其複雜程度或技術水平如何。會玩井字遊戲的電腦不用獲勝便可被認為是 AI。AI 的門檻低，不應該和人類的一般智慧相提並論。最小的一段程式碼也可以被認為是 AI，好萊塢電影在世界末日中興起的具有感知能力的機器也是 AI。

2 **人工智慧**一詞是由 John McCarthy 在 1956 年於該主題的第一次學術會議中提出。

當我們用 AI 這個詞時，它是來自於無機且一般而言就是非生物裝置的智慧的總稱。不論此詞的最低門檻為何，擁有整個研究領域和持續成長的實際應用的人類，對它都有一個統一的術語和直截了當的目標。所有量測到的東西都得到管理，因此人類開始量測、改進和競相實現更強大的 AI。

AI 的歷史

AI 的框架一開始非常的特定性，不過現今已不是如此。你或許已經知道，以 TensorFlow.js 作為框架的概念可以應用在音樂、影片、影像、統計、以及任何我們可以累積的資料。不過它並不是一開始就這樣。AI 的實作一開始只是適用於特定領域的程式碼，且缺乏任何變化的可能。

網路上有一些笑話說 AI 只是一堆 IF/THEN 敘述的集合，而我的想法是，這樣的說法並非完全是錯的。就如同我們已經提過的，AI 是所有對自然智慧的模仿的總稱。即使是新手程式設計師也會藉由解決像是 Ruby Warrior（*https://oreil.ly/ze9mi*）這樣的簡單 AI 習題來學習程式設計。這些習題會教導演算法的基礎知識，而且只需要撰寫少量的程式碼。這種簡單性的代價是，雖然它也是一種 AI，不過它只能模仿程式設計師的智慧。

長久以來，實現人工智慧的主要方法取決於編寫人工智慧程式的人的技能和哲學，它們被直接翻譯成程式碼，以讓電腦可以實行這些指令。數位邏輯會執行那些經過溝通後被寫成程式的人類邏輯。當然這就是建立 AI 時的最大遲滯。我們需要一個知道如何建立可被理解事物的電腦，而且我們會受限於它們的理解能力以及將理解進行轉譯的能力。寫死的 AI 無法推演出超越給它們的指令的事物。這大概就是將人類智慧轉譯成人工智慧的最大阻礙。如果你想要教一台電腦如何下西洋棋，你要怎麼教它西洋棋的理論呢？如果你想要告訴程式如何分辨貓和狗——這種事連小孩子都知道——你又怎麼知道演算法要怎麼寫呢？

在 1950 年代末期到 1960 年代初期，這種老師的概念從人類移轉到可以讀取原始資料的演算法。Arthur Samuel 在一個讓 AI 擺脫了其創造者的實際限制的聚會中提出了**機器學習**（*machine learning*, ML）一詞。程式可以演變以適應資料，還可以抓出程式設計師無法將其轉換為程式碼、或甚至是他們自己從不知道的概念。

以資料來訓練應用程式或程式中的函數這樣的概念是一種令人振奮的企圖心。儘管如此，在一個電腦還是整個房間大小且資料還沒有數位化的時代，這還是一個難以企及的要求。幾十年過去了，電腦已經達到了模仿人類的資訊和架構能力的臨界點。

在 21 世紀，機器學習研究者開始使用繪圖處理器（graphics processing unit, GPU）來繞過那被稱為**范紐曼瓶頸**（*von Neumann bottleneck*）的「CPU 和記憶體間的單獨通道」。2006 年時，Geoffrey Hinton 等人利用資料以及神經網路（neural networks）（我們會在下一節提到這個概念）來瞭解樣式（pattern）並讓電腦可以辨識手寫數字。對以往那些既不穩定又不完美的一般性計算方法而言，這是一種壯舉。**深度學習**（*deep learning*）可以針對手寫字的隨機性進行讀取和進行調適，並以超過 98% 的準確度識別字元。在這些發表的論文中，運用資料作為訓練者的想法從學術作品成為現實。

雖然 Hinton 受困於證明神經網路的確是可行的，那個「這是哪個數字？」的問題成為機器學習從業人員的忠貞分子。這個問題已經成為機器學習框架的典範問題。TensorFlow.js 有一個展示（*https://oreil.ly/vsANx*）可以在兩分鐘內直接在你的瀏覽器中解決這個問題。藉由 TensorFlow.js 的效益我們可以輕易的建構進階的學習演算法，並且無縫的運作在網站、伺服器、還有裝置上。不過到底這些框架實際上在做什麼呢？

AI 的終極目標永遠是趨近或甚至是超越人類在執行某一單一任務時的能力，前面說到的 98% 準確度的確做到了。Hinton 的研究引發了對這些高效能機器學習方法的關注，並創造了**深度神經網路**（*deep neural networks*）等行話。我們將在下一節詳細說明，它是應用機器學習的開始，而機器學習已開始蓬勃發展，且最終會進入像是 TensorFlow.js 等機器學習框架中。雖然基於機器的學習演算法快速的出現，它們的靈感還有術語的來源已經相當清楚。我們可以模擬我們內部的生物系統來創造某些先進的事物。從歷史上看，我們將自己和大腦皮層（cerebral cortex）（大腦的一層）當作結構化訓練和智慧的靈感泉源。

神經網路

深度神經網路的靈感來自於我們人類的身體。數位節點（node）——有時稱為**感知機網路**（*perceptron network*）——模擬我們大腦的神經元（neuron），並像我們自己的神經鍵（synapse）一樣被激發以建立思想機制。這就是為什麼這種網路被稱為**神經網路**的原因，因為它模仿了我們大腦的生化結構。許多資料科學家厭惡這種與人類大腦的類比方式，不過這樣的類比通常是適切的。藉由連結數以百萬計的節點，我們可以建立深度神經網路，一種可以進行決策的優雅數位機器。

藉由增加愈來愈多層的神經通道，我們便有了**深度學習**（*deep learning*）一詞。深度學習是隱藏的節點層的巨大分層（或深度）連結。你會聽到有人稱這些節點為**神經元**（*neuron*）、**人工神經元**（*artificial neuron*）、**單位**（*unit*）、或甚至**感知機**（*perceptron*）。這種術語上的差別證明了機器學習來自於各領域學者的貢獻。

整個學習領域只是 AI 的一部份。如果你一直在關注這個領域，人工智慧有一個稱為機器學習的子集合或分支，而深度學習的想法就位於該集合中。深度學習主要是用來增強機器學習的一種演算法類別，但它並不是唯一的一種。有關這些主要術語的直觀表示，請參見圖 1-1。

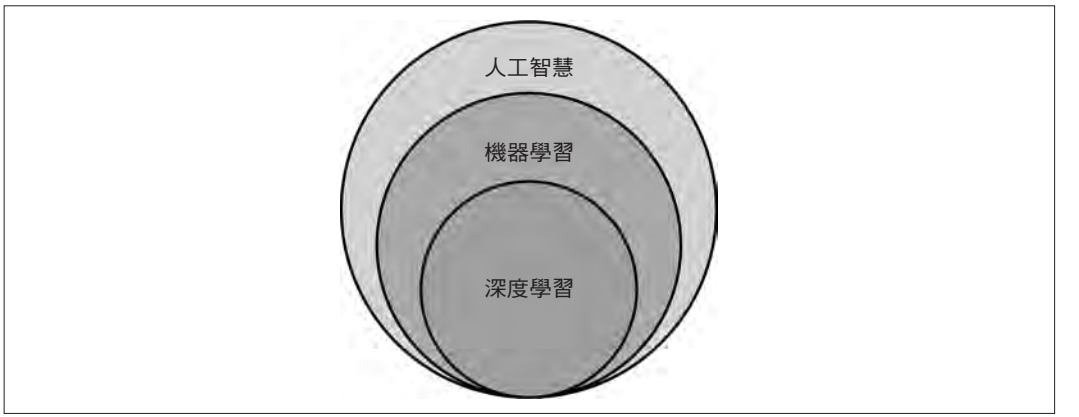

圖 1-1　AI 次領域

就像人類一樣，神經元是經由重複的以範例和資料來教導或「訓練」而適當的建立起來的。這些神經網路一開始常常做錯或隨機的表現，不過當它們一遍遍看了資料範例後，它們會「學習」到預測的能力。

不過我們的大腦並不會直接感知這個世界。就像電腦一樣，我們仰賴已被組織成相關資料送到大腦的電流信號。對於電腦而言，這些電流信號被類比成**張量**（*tensor*），我們會等到第 3 章再談這個主題。TensorFlow.js 具體呈現了研究和科學家們所證實的這類進展。這些用來幫助人體表現的技術都可以被包裝在一個優化的框架中，這樣我們就可以好好的利用這幾十年來受人體所啟發的研究。

例如，我們的視覺系統始於我們的視網膜（retina），它使用神經節（ganglion）將感光資訊傳遞到我們的大腦以激發這些神經元。你可能還記得小時候生物學裡提過，我們的視覺裡有盲點，而且技術上來說我們看到的東西是顛倒的。信號並不會「按原樣」發送到我們的大腦。這個視覺系統內建了我們在現在的軟體中所使用的技術。

雖然我們都會因為有了 8K 解析度的電視而感到興奮，你可能會認為我們的大腦和視覺仍然超越了現代的計算能力，但情況並非總是如此。將視覺信號從我們的眼睛連結到我們的大腦的線路只有大約 10 MB 的頻寬，這只相當於 1980 年代初期區域網路（LAN）的速度。即便是串流式寬頻連結也需要比這更大的頻寬。但是我們的確可以立即且快速地感知到一切事物，不是嗎？那麼訣竅是什麼呢？我們如何透過這種進階的硬體來獲得更好的信號？答案是我們的視網膜在將資料發送到我們的深度連結的神經網路前，會對資料進行壓縮和「特徵化（featurize）」。所以這就是我們即將用電腦開始要做的事情。

卷積神經網路（convolutional neural network, CNN）處理視覺資料的方式，和我們的眼睛和大腦一起壓縮和激發神經通路的方式相同。你將在第 10 章中進一步了解並編寫自己的 CNN。我們每一天都在更瞭解我們是如何運作的，並將數百萬年的進化成果直接應用到軟體中。雖然能夠瞭解這些 CNN 的工作原理對你來說是很棒的事，但要我們自己編寫它們有點太學術性了。TensorFlow.js 包含了你在處理影像時所需要的卷積層。這是利用機器學習框架的基本好處。

你可以花費數年時間閱讀和研究那些可以讓電腦視覺、神經網路和人類有效運作的所有獨特技巧和竅門。但在我們目前所處的時代中，這些根基已經有時間來成長、分支並最終結出果實：這些先進的概念已經內建至我們周圍的服務和設備中來提供使用。

今日的 AI

現在我們會用這些 AI 的最佳實踐作法來增強機器學習的能力。用來進行邊緣偵測（edge detection）、關注特定區域、甚至單一物件的多相機輸入的卷積層為我們提供了一個預先消化過的資料金礦，而它是透過光纖雲端機器伺服器群進行 AI 訓練的。

2015 年時 AI 演算法在某些視覺任務的效能上開始超越人類。你可能在新聞中看過，AI 已經在癌症偵測（*https://oreil.ly/ZCz0B*）上超越人類的表現，甚至在識別法律缺陷方面的表現上也優於美國頂級律師（*https://oreil.ly/9dW3S*）。藉由使用數位化資訊，AI 可在幾秒鐘內完成這項工作，而不是幾小時。AI 的「魔法」實在令人敬畏。

人們正在尋找如何應用 AI 在他們的專案，甚至是建立全新產業的嶄新和有用方法。

AI 已被應用於：

- 在寫作、音樂和視覺方面產生新的內容
- 推薦有用的內容
- 取代簡單的統計模型
- 由資料中推演出規律
- 分類器和識別器視覺化

所有這些突破都是在深度學習的層面。今天，我們擁有必要的硬體、軟體和資料，可以透過深度機器學習網路實現突破性的變革。網路社群和《財星》500 大公司每天都在發佈 AI 領域的新資料集、服務和架構上的突破。

你可以運用你可用的工具以及本書中的知識來輕鬆建立前所未見的事物並將它們帶進網路。無論是為了娛樂、科學還是財富，你都可以為任何現實世界的問題或業務建立可擴展的智慧型解決方案。

機器學習當前可能存在的問題是它已經是一個新的超級強權，而且涵蓋面很廣。我們沒有足夠的範例來理解在 JavaScript 中使用 AI 的全部好處。當電池的使用壽命被顯著延長時，它們開啟了一個全新的裝置世界，從功能更強大的手機到一次充電可以使用數個月的相機。這一突破在短短幾年內就為市場帶來了無數的新產品。機器學習不斷取得突破，它以指數級的速度留下了一堆我們甚至無法理解或識別的新技術進展。本書將以適當的方式聚焦於具體和抽象的範例，以讓你將 TensorFlow.js 應用於實務的解決方案。

為何要使用 TensorFlow.js？

你是有選擇的。你可以從頭開始編寫自己的機器學習模型，也可以從各種程式語言的任何現有框架中進行選擇。即使在 JavaScript 領域，也已經存在相互競爭的框架、範例、以及選項。是什麼讓 TensorFlow.js 能夠處理和承載當今的 AI？

大力支持

TensorFlow.js 是由 Google 建立和維護的。我們將在第 2 章中詳細介紹這一點，但值得注意的是，世界上一些最優秀的開發人員齊心協力使 TensorFlow.js 成為現實。這也意味著，無需社群的任何付出，TensorFlow.js 就能夠運用最新和最偉大的突破性發展。

與其他基於 JavaScript 的機器學習程式庫和框架的實作方式不同，TensorFlow.js 支援經過優化和測試後的 GPU 加速程式碼。這種優化會讓你以及你的機器學習專案受益。

上線就緒

大多數機器學習解決方案僅限用於高度客製化的機器。如果你想建立一個網站來分享你的突破性技術，那麼 AI 通常被隱藏在 API 之後。雖然這對於在 Node.js 中執行的 TensorFlow.js 來說是完全可行的，但對於直接在瀏覽器中執行的 TensorFlow.js 亦可行。這種免安裝體驗在機器學習領域頗為罕見，它使你能夠無礙地分享你的創作。你可以存取互動性的環境並進行版本控制。

離線就緒

JavaScript 的另一個好處是它可以在任何地方執行。程式碼可以像漸進式 web 應用程式（progressive web app, PWA）、Electron 或 React Native 應用程式一樣儲存到使用者的裝置上，而後便可以在沒有任何網路連結的情況下執行。不用說，與託管（hosted） AI 解決方案相比，這也顯著提高了速度和成本效益。在本書中，你將發現無數完全存在於瀏覽器上的範例，這些範例使你和你的使用者降低了遲滯（latency）延遲和託管成本。

隱私性

AI 可以幫助使用者識別疾病、稅務異常和其他個人資訊。透過網路發送敏感資訊可能很危險。裝置上的結果就應該留在裝置上。你甚至可以在不將資訊傳到瀏覽器外的狀況下訓練 AI，並將結果儲存在使用者的機器上。

多樣性

已被應用的 TensorFlow.js 在機器學習領域和平台上造成巨大而廣泛的影響。TensorFlow.js 可以善用更強大的機器中的 CPU 或 GPU 來執行 Web Assembly。對於新進者來說，當今具有機器學習的 AI 是一個包含了新術語和複雜性的重要又廣闊的世界，擁有一個可以處理各種資料的框架很有用，因為它可以讓你的選擇性保持多元。

掌握 TensorFlow.js 可以讓你將技能應用到支援 JavaScript 的各種平台（參見圖 1-2）。

圖 1-2　TensorFlow.js 平台

使用 TensorFlow.js，你可以自由選擇、建構原型並將你的技能部署到各個領域。為了充分利用你在機器學習上的自由性，你需要瞭解一些可以幫助你進入機器學習領域的術語。

機器學習類型

很多人將機器學習分為三類，但我相信我們需要用四個重要元素來看待機器學習：

- 監督式（supervised）
- 非監督式（unsupervised）
- 半監督式（semisupervised）
- 增強式（reinforcement）

這每一個元素都值得大書特書。下面的簡短定義只是簡單的參考，可讓你熟悉在該領域中會聽到的術語。

快速定義：監督式學習

在本書中，我們將聚焦於最常見的機器學習類別，即**監督式機器學習**（*supervised machine learning*）（有時稱為**監督式學習**（*supervised learning*）或簡稱為**監督式**（*supervised*））。監督式機器學習只是意味著我們對用來訓練機器的每個問題都會有一個答案。也就是說，我們的資料是有標籤的（labeled），因此，如果試圖教導機器去分辨照片裡是否包含鳥類，我們可以立即對 AI 進行判斷其對錯。就像答案卡掃描機（Scantron）一樣，我們會有標準答案。但與答案卡掃描機不同的是我們還可以識別答案的錯誤程度。

如果 AI 有 90% 的把握確定一張鳥的照片裡就是一隻鳥，即使它的答案是正確的，還是可以有 10% 的改善空間。這闡明了 AI 透過立即滿足資料來「訓練」的層面。

如果你沒有幾百個已標記的問題和答案的話，還請不要擔心。在本書中，我們不是會為你提供已標記資料，就是會向你展示如何自己產生已標記資料。

快速定義：非監督式學習

非監督式學習不需要我們先有答案。我們只需要問題就夠了。非監督式機器學習應該是比較理想的，因為世界上大多數的資訊都沒有標籤。這類機器學習側重於機器可以從未標記的資料中學習和彙整它們的內容。雖然這個主題看起來可能有點令人困惑，但其實人類每天都在做這件事！例如，如果我給你一張我花園的照片，並問你我擁有多少種不同類型的植物，你不必知道每種植物的屬和種就可以告訴我答案。這有點像我們如何理解我們自己的世界。許多非監督式學習都專注於對大量資料進行分類以供使用。

快速定義：半監督式學習

大多數情況下，我們的資料不會是 100% 未標記的。回到之前的花園範例，你可能不知道每種植物的屬和種，但也並非完全無法將植物分類為 A 和 B。你可能會告訴我說我有十株植物，其中包含了三朵花和七株藥草。長久以來都只有少量已標籤的資料，所以目前半監督式學習的研究非常火熱！

你可能聽說過**生成網路**（*generative network*）或**生成對抗網路**（*generative adversarial network*, GAN）這些術語。許多 AI 新聞文章中都提到了這些流行的 AI 架構，而它們源自於半監督式學習策略。生成網路根據我們希望網路建立的範例進行訓練，並透過半監督式方法建構新範例。生成網路非常擅長從小型已標記資料集來建立新內容。流行的 GAN 案例通常有自己的網站，例如越來越受歡迎的 *https://thispersondoesnotexist.com*，創意人員正享受著這種半監督式的輸出結果呢。

GAN 在生成新內容方面發揮了重要作用。雖然流行的 GAN 是半監督式的，但 GAN 的更深層概念並不限於半監督式網路。人們已經將 GAN 進行修改以處理我們所定義的每種學習類型。

快速定義：增強式學習

解釋增強式學習的最簡單方法是透過處理現實世界中的活動來證明它是必要的，而不是像之前一樣的學理說明。

例如如果我們正在下西洋棋，一開始我移動了士兵，這是好棋還是壞棋？或者，如果我想讓一個機器人將球踢進球門，一開始它向前邁出一步，那是好是壞？就像人類一樣，答案取決於結果。為了獲得最大回報經常要進行一系列動作，而不總是只用單一動作來產生單一的結果，訓練機器人要先走一步或先看一下是很重要的，但可能不像它在其他關鍵時刻所做的事那麼重要。而那些關鍵時刻都以獎勵（reward）來進行強化。

如果我教 AI 玩**超級馬里歐兄弟**（*Super Mario Bros.*），我想要高分還是快速過關？獎勵會告訴 AI 什麼動作組合可以將目標最佳化。增強式學習（RL）是一個不斷擴展的領域，經常與其他形式的 AI 結合以培育出最好的結果。

資訊過載

對剛剛所提到的機器學習應用的數量感到驚訝是合理的。在某種程度上，這就是為何我們需要像 TensorFlow.js 這樣的框架的原因。我們甚至無法理解這些奇妙系統的所有用途及其對未來幾十年的影響！當我們圍繞這一點思考時，AI 和 ML 的時代已經到來，而我們將成為其中的一部分。監督式學習是實現 AI 所有優勢的重要第一步。

我們將介紹機器學習的一些最令人興奮但又實際的用途。在某些層面上我們只會淺嚐即止，而在其他層面，我們將深入研究它們的工作原理。以下是我們將涵蓋的一些廣泛類別。這些都是監督式學習的概念：

- 影像分類（image ccategorization）
- 自然語言處理（natural language processing, NLP）
- 影像切割（image segmentation）

本書的主要目標之一是，讓你可以理解類別的概念，卻不會因而受限。我們將傾向於運用實驗和實踐科學。有些問題可以透過過度工程（overengineering）解決，有些問題可以透過資料工程（data engineering）解決。使用 TensorFlow.js 來查看、識別和建立新工具的關鍵就是用 AI 和機器學習來思考。

無所不在的 AI

我們正進入 AI 正在滲透一切的世界。我們的手機現在加速升級成為深度學習硬體，攝影機正在應用即時 AI 偵測。在撰寫本文時，一些無人駕駛的汽車正行駛在我們的街道上。

在過去的一年裡，我甚至注意到我的電子郵件開始使用「按 tab 來完成（tab to complete）」選項來完成我的句子。其實我不太想承認，這個功能的結果比我原來寫的東西更清晰、更簡潔。這是一項顯著的成就，它使得多年來一直在收件箱中保護我們免受垃圾郵件侵害的那個被遺忘的機器學習 AI 黯然失色。

隨著這些機器學習計畫的展開，新平台的需求越來越大。我們正在一步步的將模型導入手機、瀏覽器和硬體等邊緣裝置上。我們正在尋找新的語言來達成任務是有道理的，這個搜尋很快就會因為得到 JavaScript 這個選項而結束。

框架提供的內容之旅

機器學習到底是什麼？以下是一個會讓博士班學生因其簡潔而敬畏的準確描述。

一般的程式碼是由人類直接編寫後，再由電腦進行讀取並解譯此程式碼或其衍生物。現在我們處於一個人類不再編寫演算法的世界，那麼真實的情況如何？演算法又從何而來？

這只需要再多走一步——由人來寫出演算法的訓練器。藉由框架的幫助，或甚至完全從頭開始，人類在程式碼中概述問題的參數、所需的結構以及要學習的資料所在的位置。現在機器執行這個訓練寫程式的程式，它會不斷的編寫一個會不斷改進的演算法作為該

問題的解決方案。在某個時候，你可以停止這個程式、取出它所寫的最新演算法結果並且使用它。

就這樣！

此演算法會比用於建立它的資料小多了。大小數以 GB 計的電影和影像被用來訓練機器學習解決方案好幾個星期，這些都只是為了建立大小只有幾 MB 的資料來解決一個非常特定的問題。

產生的演算法本質上就是一組數值，這些數值可以滿足人類程式設計師所指定的輪廓結構。這些數值及其相關的神經圖的集合通常稱為**模型**（*model*）。

你可能在瀏覽技術論文時看過這些圖，它們被畫成從左到右的節點集合，如圖 1-3 所示。

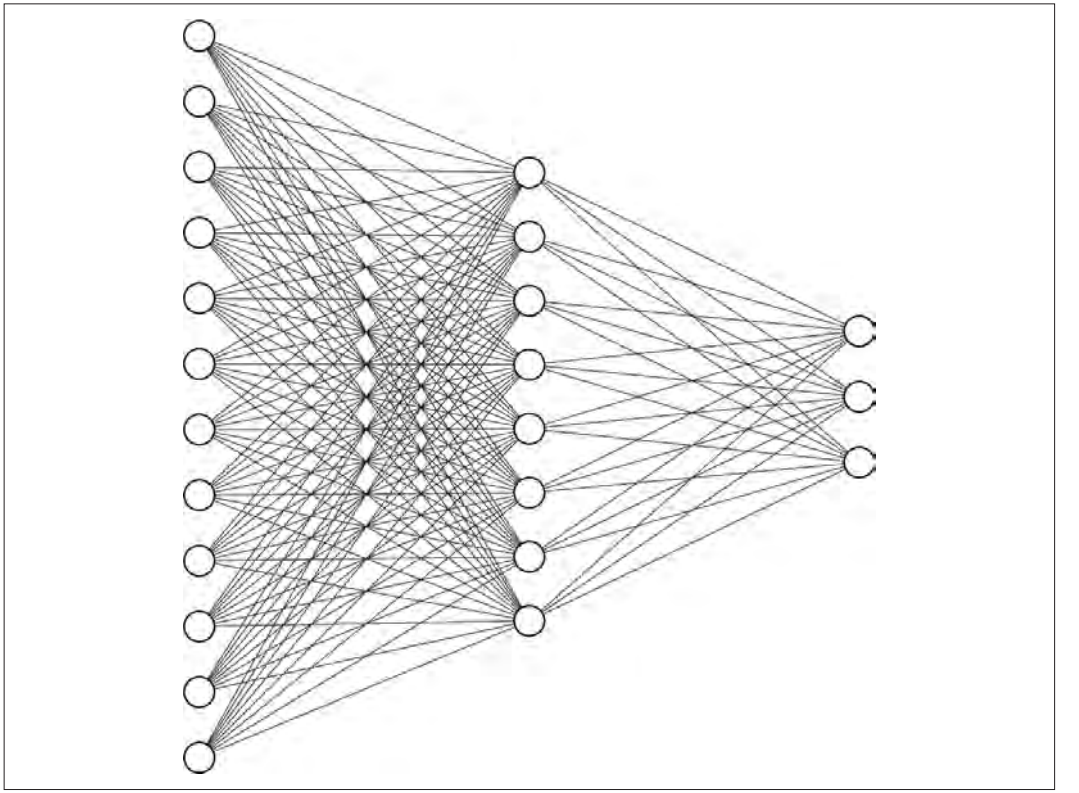

圖 1-3　密集連結神經網路範例

我們的框架 TensorFlow.js 會處理一些 API，用來指定模型的結構或架構、載入資料、傳遞資料給我們的機器學習程序、以及調整機器以能得到更好的預測結果。這就是 TensorFlow.js 的真正優勢所在。我們所需要擔心的只有如何使用足夠的資料並適當地調整框架來解決問題，還有儲存產生出來的模型。

模型是什麼？

當你在 TensorFlow.js 中建立神經網路時，它把所需的神經網路用程式碼來表示。該框架會產生一張圖，其中的每個神經元都設定為精心設計的隨機值。此時模型的檔案大小一般而言是固定的，但其內容會不斷變化。當資料透過這個未經訓練的隨機值網路來進行預測時，我們得到的答案通常會和正確答案相差甚遠，和亂猜一樣。我們的模型沒有對任何資料進行訓練，所以它的表現很糟糕。所以身為開發人員，我們所編寫的程式碼是完整無誤的，但沒有訓練的話結果很差。

一旦訓練迭代（iteration）進行了相當時間後，神經網路中的權重就會被評估並進行調整。速度，通常稱為**學習率**（*learning rate*），會影響最終產出的解決方案。以特定學習率進行數千次這種小步驟之後，我們會看到一台有所改進的機器，而我們正在設計一個成功機率遠遠超過原始機器的模型。網路已經不再是隨機的，並會收斂於那些使神經網路有效用的數值！指定給那個給定結構中神經元的那些數值就是訓練後的模型。

TensorFlow.js 知道如何記錄這些數值和圖結構，因此我們不必這樣做，而且它也知道如何以適當且好用的格式來儲存這些資訊。

一旦有了這些數值，我們的神經網路模型就可以停止訓練，而僅用於進行預測。用程式設計術語來說，它已經成為一個簡單的函數。傳入資料，傳出結果。

將資料餵入神經網路看起來很像碰運氣，如圖 1-4 所示，但在計算世界中，它是一種在機率和分類中取得平衡的精密機器，可產生具有一致性和可重複性的結果。資料被輸入機器中，然後出現一個機率性的結果。

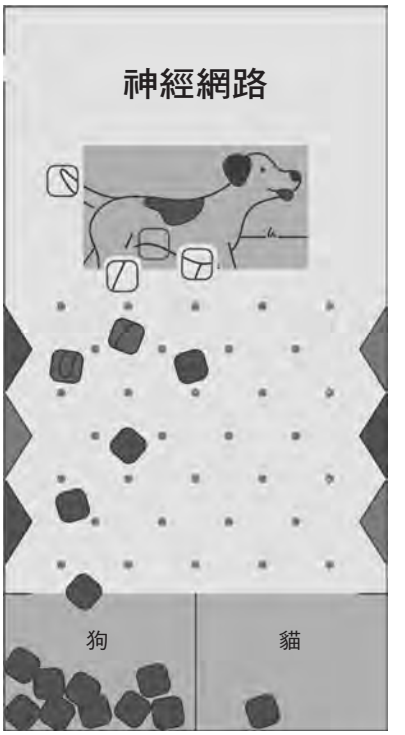

圖 1-4　平衡網路的一種隱喻表達

在下一章中，我們將嘗試匯入經過完全訓練的模型並進行預測。我們將利用花費數小時訓練所獲得的威力，在幾微秒內獲得智能性的分析結果。

本書內容

這本書的建構是為了讓你可以把它打包去度假，一旦你找到了自己的那片天堂後，就可以閱讀本書、學習概念並複習答案。影像和螢幕截圖應該足以解釋 TensorFlow.js 的深入資訊。