《邊緣 AI》精彩推薦

本書以實用且易懂的方式，向讀者介紹新興且快速成長的邊緣 AI 領域。它揭開了專業術語的重重迷霧，並點出了在建置邊緣 AI 應用時可能遇到的真實挑戰。本書提供了從概念到部署的必備指南，對於初入此領域的讀者來說是必讀之作。

— *Wiebke Hutiri*，荷蘭台夫特理工大學（Delft University of Technology）

我超愛這種能把複雜的技術主題講解得淺顯易懂的寫作風格。我可以想像大家把這本書當做參考書反覆查找，至少我自己就會這樣做！

— *Fran Baker*，永續與社會影響力部門主任，Arm

本書以極具啟發性和全面性的方式，介紹邊緣 AI 這個新興領域！本書涵蓋的主題非常廣泛，從核心概念一路講到最新的軟硬體工具，不但提供了實用建議，還包含了多個端對端範例。任何剛踏入這個新興領域的人，都會受益於這本書所提供的深刻見解和清晰思緒。

— *Aurélien Geron*，曾任 YouTube 自動影片分類小組主管與暢銷書作家

這是一本建立智慧裝置的指南書：完整介紹結合現今 AI 智慧技術和嵌入式系統的方法。

— *Elecia White*，《Making Embedded Systems》作者
與《Embedded 》數位廣播節目主持人

推薦序

Thomas Dohmke（GitHub CEO）在 2022 年曾說：「我認為轉向雲端的趨勢會迅猛到一個程度，您的本地電腦上預計在短短幾年內將不再有任何程式碼[1]。」然而，這本書充分說明了為什麼我和許多其他身處這個新興邊緣機器學習領域的人一樣，認為他可是大錯特錯。

我們開始看到許多像是高品質語音辨識、森林防火、智慧居家控制等實際應用紛紛出籠，這些應用能完全實現是因為本地裝置現在已可執行各種高階的機器學習演算法。Jenny 和 Dan 寫了一本非常精彩的書，不僅解釋了為什麼在邊緣應用中加入智慧性功能（智能）是解決重要問題的關鍵點，還帶領讀者一步步了解設計、實作和測試這類應用所需的各個步驟。

當您第一次開始研究邊緣機器學習專案時，可能覺得它相當嚇人。這個領域涉及很多術語、變化超快，還需要嵌入式系統和 AI 等領域的知識，這些領域以往整合得不算太好。本書作者的成就在於，他們用輕鬆而全面的方式介紹能讓應用程式有效運作所需了解的所有內容。多虧了他們極力強調的各種真實世界案例，並且即便是複雜的主題也是用一般口語而非數學或程式碼來解釋，都讓本書更適合推薦給產品經理、高階主管、設計師以及工程師。

1　來源：GitHub 的 X（前 Twitter）帳號頁面：*https://oreil.ly/WgTQu*。

他們成功地把許多從經驗中獲得的艱苦知識精煉成了課程，這給了所有要開發這類應用的團隊一個非常好的起點。

本書也成功跨越建立邊緣機器學習應用的實際考量，並幫助您了解如何避免在工作中造成傷害。圍繞著 AI 的各種道德問題似乎令人不知所措，但本書作者將它們拆解成您可以直接應用於專案規劃和測試過程中的各種問題點。這有助於專案的所有利益相關者彼此協作，並希望能夠避免當電腦在我們的生活中具備更多決策權力時所涉及的諸多潛在危險。

我從事邊緣機器學習應用開發已經十多年了，首先在一家新創公司，然後在 Google 擔任技術主管，現在是另一家新創公司的創辦人，我會要求所有加入本團隊的夥伴都要閱讀本書。如果您對這個領域的任何面向感興趣的話，無論是作為開發者、設計師、經理，或只是關心這項新興科技，我都極力推薦這本書。我保證本書會給您許多迷人的想法，並幫助您做出新一代的智慧型裝置。

— *Pete Warden，Useful Sensors 公司 CEO，*
TensorFlow Lite for Microcontrollers 負責人

中文版作者序

Dear Reader,

Edge AI is a tool we can use to improve our world. We wrote AI at the Edge to help teams and individuals find success with this technology, building products that make a positive impact across diverse industries and fields.

With this in mind, we're incredibly happy to share the Traditional Chinese edition of the book. We know this translation will be read by many brilliant, visionary people who will go on to build wonderful things.

In the time since the original edition was published, AI has become the world's biggest topic of conversation—and edge AI has entered public awareness. Teams around the world are creating innovative products using the technologies we describe in this book.

With this new edition, we invite you to join them. We hope you feel empowered to imagine, design, and build systems that were not possible a few years ago—and help create a better world.

We'd like to extend our warmest gratitude to Dr. David Tseng from CAVEDU Education for the translation you are reading. David is a passionate ambassador for edge AI, and it's been a privilege to work with him on this project.

We hope you enjoy the book!

Sincerely,

Daniel Situnayake & Jenny Plunkett

第一章

邊緣 AI 簡介

歡迎加入我們！本章將進行一趟邊緣 AI 的深度之旅。我們將定義許多關鍵詞彙、學習「邊緣 AI」如何不同於其他 AI 應用，並探索一些最重要的使用案例。本章的目標是回答這兩個重要問題：

- 到底什麼是邊緣 AI ？
- 為什麼我會需要它？

定義關鍵詞彙

每個技術領域都有自己的流行用語，邊緣 AI 也不例外。實際上，*邊緣 AI* 一詞是兩個流行用語所組合成的一個強大術語，常常與同類詞，例如*嵌入式機器學習*和 *TinyML* 一起使用。在往下以前，最好花些時間定義這些詞彙，並了解它們的含義。由於所看到的是複合式流行語，因此讓我們先處理最基本的部分。

嵌入式

什麼是「嵌入式（embedded）」？根據您的背景，這可能是我們試圖描述的所有術語中最讓人耳熟能詳的一個。*嵌入式系統*是指可控制各種實體裝置內部電路的電腦，從藍牙耳機到最新汽車的引擎控制單元。嵌入式軟體則是運行於其上的軟體。圖 1-1 是一些可以找到嵌入式系統的地方。

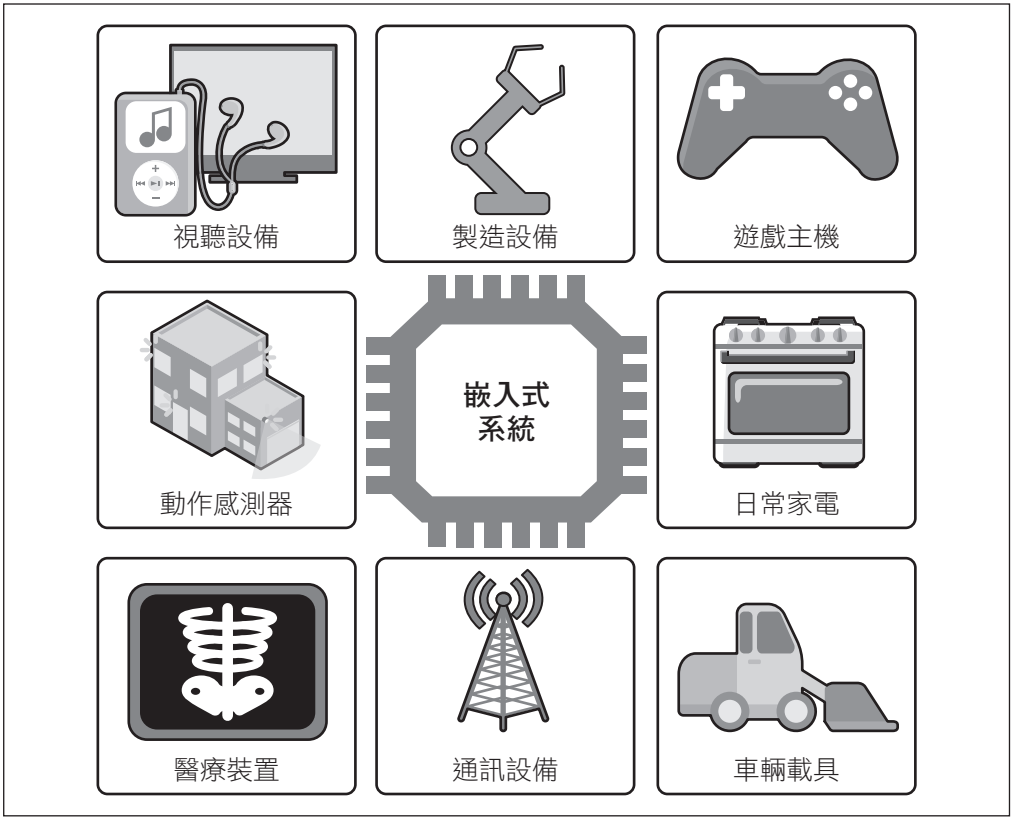

圖 1-1　嵌入式系統存在於我們世界的每個角落，包括家庭和工作場所

嵌入式系統可以非常迷您又簡潔，例如控制數位手錶的微控制器，但也可能既龐大又複雜，例如智慧電視中的嵌入式 Linux 電腦。相較於通用型電腦（例如筆記型電腦或智慧型手機），嵌入式系統通常是用於執行一個特定的專門任務。由於現代科技的大部分都是由它們所驅動，嵌入式系統非常普及。事實上，2020 年的微控制器出貨量就超過了 280 億個[1]，這還只是諸多嵌入式處理器的其中一種而已。這些裝置存在於我們的家庭、車輛、工廠和都市街道中。您和某個嵌入式系統的距離可能一直以來都只有幾英尺而已。

嵌入式系統通常都反映了它們所部署的環境中的各種限制。例如，許多嵌入式系統需要以電池電源來運作，因此它們在設計時就必須考慮到能源效率，但這也會

1　資料來源：Business Wire（*https://oreil.ly/xa0o-*）。

導致記憶體或時脈受限。而開發嵌入式系統程式則是在這些限制中自在遨遊的藝術，寫出能夠執行所需任務的軟體，同時還能夠充分利用有限的資源，這真的非常不容易。嵌入式系統工程師是現今世界的無名英雄。如果您碰巧是其中之一，感謝您的大力付出！

邊緣（與物聯網）

電腦網路的歷史是一場浩大的拔河賽。最初的電腦系統，事實上是一台可塞滿整個房間的電腦，其運算在本質上是集中式的。只有一台機器，而這一台機器必須完成所有工作。然而，電腦逐漸可以連接終端機（如圖 1-2），並讓終端機去分攤一些責任。大部分的運算都是在中央主機中進行的，但一些簡單的任務，像是如何在 CRT（陰極射線管）螢幕上渲染各種字母，則需要透過終端機的電子裝置來完成。

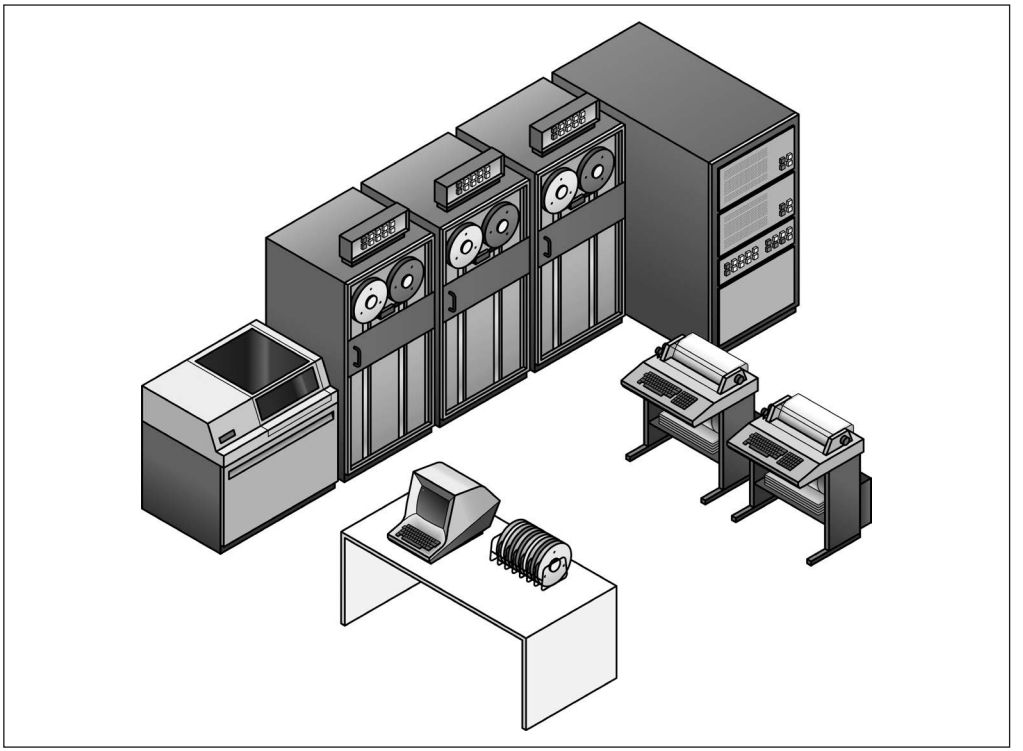

圖 1-2　大型主機負責了絕大部分的運算，而功能簡易的終端機則負責處理輸入、顯示輸出與渲染各種基本圖像

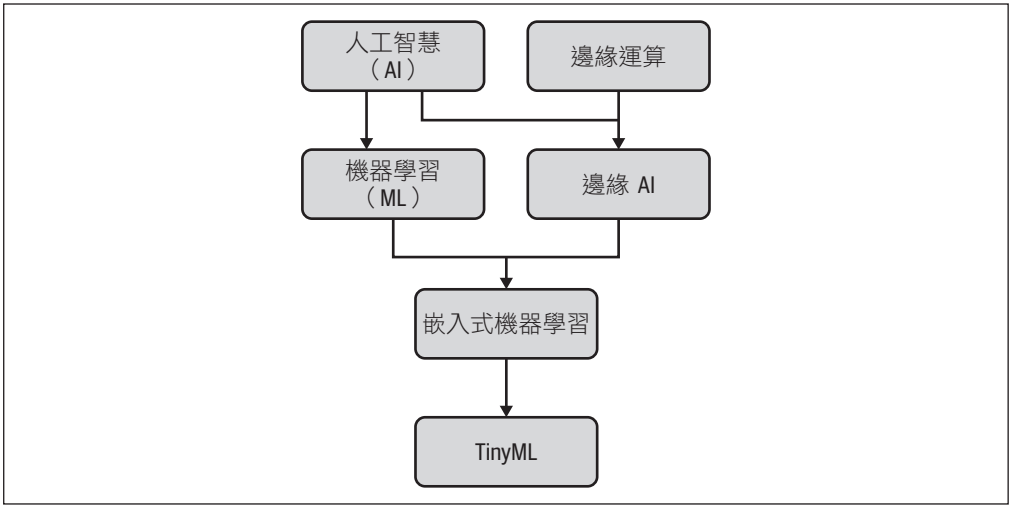

圖 1-8　本圖說明了邊緣 AI 中一些最重要的概念，從最上面的一般化名詞到最下面的最具體名詞

下一段就會深入探討邊緣 AI 這個主題，並開始分析它之所以成為一項重要技術的原因。

為什麼需要邊緣 AI ？

假設今天早上您在約書亞樹（Joshua Tree）國家公園跑步，這裡是位於美國南加州沙漠中一望無際的曠野。您整路都在聽著透過毫無間斷的手機資料連線串流而來的音樂。到了深山裡的某個超美景點，您拍了一張照片並傳送給另一半，幾分鐘後就收到了回覆。

身處在一個即使是與世隔絕之處都能有某種形式的資料連線世界中，為什麼還需要邊緣 AI ？如果強大的網路伺服器只有一步之遙，那麼能夠自行決策的小型裝置還有什麼意義？加進這些額外的複雜考量之後，難道不會讓自己的生活變得更加艱困嗎？

正如您可能猜到的那樣，答案是否定的！邊緣 AI 解決了一些非常實際的問題，否則這些問題將阻礙那些增進人類福祉的技術持續發展。我們最喜歡用來解釋邊緣 AI 優點的框架是一個聽起來粗魯但很好記的詞：BLERP。

運用 BLERP 口訣來了解邊緣 AI 的好處

BLERP 是什麼意思？ Edge AI and Vision Alliance 公司創辦人 Jeff Bier 推出了這款出色的工具（*https://oreil.ly/UY-DG*），並充分表達邊緣 AI 的好處。它包含了五個詞：

- 頻寬（Bandwidth ）
- 延遲（Latency）
- 經濟效益（Economics）
- 可靠度（Reliability）
- 隱私（Privacy）

掌握 BLERP 口訣之後，任何人都可以輕鬆記住並說明邊緣 AI 的優點。它也可以作為一個篩選器，幫助您決定邊緣 AI 是否適用於某個特定應用。

現在來一一介紹它們吧。

頻寬

物聯網裝置擷取的資料量通常會超過它們的傳輸頻寬。這代表它們所擷取的絕大多數感測器資料甚至還沒使用就被丟掉了！想像一下，有一個用於監測工業機台振動情形的智慧感測器，可透過簡單的門檻值演算法來理解機器的振動何時太大或太小，然後藉由低頻寬連線來傳送這些資訊，以判斷機台是否正常運作。

這聽起來已經很有用了。但如果您可以辨識出資料中的不同樣式，而這些樣式可以提供機器是否即將故障的線索，聽起來如何？只要頻寬充足，我們確實可以把感測器資料發送到雲端並分析，藉此了解是否即將發生故障。

然而在許多情況下，可動用的頻寬或能源預算都不足以持續地將大量資料流發送到雲端。這代表大部分的感測器資料只能被迫丟棄，即使它包含的訊號再有用也一樣。

頻寬受限是非常普遍的狀況。這不只與可用的連線速度有關——還牽涉到能源問題。網路通訊一般來說都是嵌入式系統所執行任務中最耗電的一項，代表限制因素往往來自於電池壽命。某些機器學習模型可能需要大量運算，但相較於傳輸訊號所需的能源其實還是比較少的。

邊緣 AI 這時候就登場啦。如果物聯網裝置本身就能分析資料而無需上傳，事情會不會不一樣？這樣一來，如果分析結果顯示機器即將故障，我們依然可透過有限的頻寬來發送通知。這會比把所有資料串流出去要可行多了。

當然，裝置完全沒有（或無法進行）網路連線功能也很常見！在這種情況下，邊緣 AI 會讓以往根本不可行的諸多案例成為可能。稍後就會深入探討。

延遲

傳輸資料需要時間。就算可用的頻寬超級充足，從裝置到網際網路伺服器來回還是需要數十或數百毫秒。某些情況下的延遲會高達數分鐘、數小時或甚至幾天——想一下那些衛星通訊或儲存 / 轉發通訊就知道了。

某些應用還需要更快有所回應。例如，使用遠端伺服器來控制移動中的車輛可能是不切實際的。控制車輛在環境中導航時，需要不斷回饋方向盤轉動程度和車輛位置。在高延遲的情況下，控制方向盤轉多少變成了一個大挑戰！

邊緣 AI 藉由完全消除來回時間來解決這個問題，自駕車就是最好的例子。車輛的 AI 系統是運行於車載電腦上，這使得它能夠對各種變化快速做出反應，例如前方駕駛猛踩煞車。

邊緣 AI 作為克服延遲的武器，最具吸引力的例子之一就是機器人太空探索。火星距離地球非常遠，即使以光速傳輸也需要數分鐘的時間才能到達。更糟糕的是，由於行星的排列方式，直接通訊通常是不可能的。這使得控制火星探測車非常困難。美國國家航空和太空總署（NASA）就透過邊緣 AI 來解決這個問題，他們的探測車使用非常先進的 AI 系統（*https://oreil.ly/iQr8t*）來規劃任務、環境導航，並在另一個世界的地表尋找生命。如果您有空的話，甚至可以通過標註資料來改善演算法，藉此幫助火星探測車日後的導航任務（*https://oreil.ly/RATTg*）！

經濟效益

建立網路連線需要花很多錢。連網產品使用起來自然更為昂貴，且其所仰賴的基礎建設也需要製造商投入資金。所需頻寬越多，成本就越高。對於部署於天涯海角而需要通過衛星進行長距離連線的遠端裝置來說，情況尤其糟糕。

透過在裝置端直接處理資料，邊緣 AI 系統減少或避免了以網路傳輸資料和在雲端處理資料的成本。這使得許多以往無法實現的案例變為可能。

在某些情況下，唯一可行的「連線」方式是派出一批人去親自執行任務。例如，保育研究人員通常會使用隱藏式相機在偏遠地區中監測野生動物。這些裝置會在偵測到動作時拍攝照片，並將其儲存到 SD 卡中。由於透過衛星網路來上傳所有照片的成本實在太高，因此研究人員必須親自前往相機安置地蒐集影像，並清除儲存空間。

傳統的隱藏式攝影機因為一有動作就會觸發，不管是風吹樹枝還是登山者經過等，所以會拍下許多無關緊要，或研究人員不感興趣的生物照片。但是現在有些團隊已運用邊緣 AI 來辨識出他們感興趣的動物，其他無關的影像就能直接捨棄。這意味著他們不再需要「那麼」頻繁地進入深山荒野去更換 SD 卡了。

在其他情況下，連線成本可能不是問題。然而，對於依賴伺服器端來提供 AI 功能的產品來說，維護伺服器基礎架構的成本可能會使您的商業模式更為複雜。如果您必須支援一批需要「打電話回家」才能做出決策的裝置，就可能會被迫採用訂閱模式。您還必須承諾長期維護伺服器——冒著顧客因為您決定不再支援而讓裝置「變磚」的風險[7]。

不要低估經濟效益的影響力。藉由降低長期支援成本，邊緣 AI 實現了許多以往視為不可行的應用案例。

可靠度

由裝置端 AI 所控制的系統有機會比那些仰賴雲端連線的系統更可靠。當您為裝置加入無線連網功能時，實際上是加入了一個巨大且高度複雜的相依網路，包含了連接層通訊技術，到運行您應用程式的網路伺服器。

這團謎霧的許多地方都超出了您的可控範圍，因此即使所有決定都是正確的，還是會面臨與您所用的分散式運算堆疊相關的可靠度風險。

---

7 由於有必要去監控裝置並推送演算法更新，因此並非所有的邊緣 AI 都能躲過這個狀況。儘管如此，許多情況下，邊緣 AI 的確可因此減輕維護負擔。

第九章

開發邊緣 AI 應用程式

開發邊緣 AI 應用程式是一項龐大的任務。本章我們將進一步認識迭代式開發模型，這有助於在真實世界的專案中成功地部署邊緣 AI。

邊緣 AI 的迭代式開發工作流程

開發一個成功的應用程式的步驟其實很簡單：從小處著手、逐步修改、測量進度，並在達成目標後停止。複雜性會在引進組成邊緣 AI 技術的大量浮動成分時隨之而來。本章的目標在於提供具體的開發步驟，讓您可以盡可能提高成功的機會。

如同第 177 頁「邊緣 AI 工作流程」所述，這個工作流程的核心來自於回饋循環的動力。我們的目標是在過程中的各個階段之間建立回饋循環，從而不斷地改善對問題、解決方案以及最佳整合方式的理解，如圖 9-1。

雖然這是一個迭代過程，有些部分仍比其他部分更具迭代性。一些之前探討過的步驟—探索、目標設定和引導—用來確定我們想做什麼以及如何進行。它們首先出現在前期規劃，當有新訊息加入時也會再次出現於定期的重新評估：也許是在初步部署之後，又或者是在浮出大量新資料時。

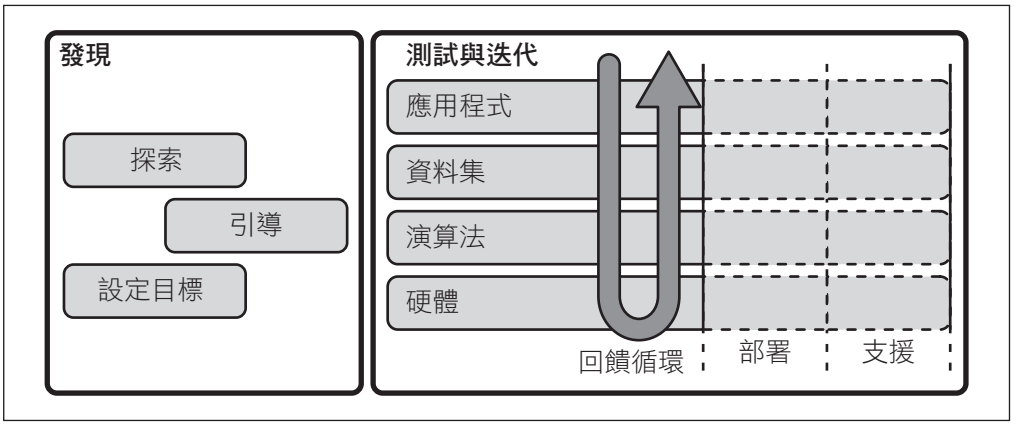

圖 9-1　回饋循環是邊緣 AI 工作流程的核心，這個概念首次出現於本書第 177 頁「邊緣 AI 工作流程」中

工作流程中段的*測試和迭代*部分會有更大幅度的迭代。它們是開發、測試、改良緊密螺旋的一部分，旨在達到您設定的任何目標。您可以將它們視為四條平行的開發軌道，在為了滿足需求而改進的過程當中也會互相交流。

部署和支援也是迭代的，但相較於核心來得慢了些。這是它們的特性之一：一旦部署並交到使用者手中，系統開發必定會減緩。然而，這個階段將開始獲得至關重要的回饋，也是系統不得不開始適應不斷變化的真實環境的時候。能夠越早部署並挖掘這條觀察脈絡越好。

下一段將逐步說明工作流程中的每個主題，並詳細介紹關鍵活動與概念。

探索

探索幫助我們了解想要做什麼。它包括了在第六章中所學的大部分工作，並包含以下主要任務：

- 描述想要解決的問題（第 180 頁「描述問題」）
- 確定是否需要使用邊緣 AI（第 181 頁「我需要部署到邊緣嗎？」以及第 187 頁「我需要用到機器學習嗎？」）
- 確定專案是否可行（第 195 頁「確定可行性」）

- 把問題映射到已知的方法論來定義問題（第 205 頁「界定問題」）
- 分析解決方案的潛在風險、傷害和意外後果（第 196 頁「道德可行性」）
- 列出利益關係者並了解他們的需求（第 135 頁「利益關係者」）
- 進行初步的資料探索

最後一步很大程度上取決於現階段是否已能夠取得資料集，即使數量有限。強烈建議您在嘗試確定可行性時就要掌握一些資料：資料可以清楚呈現一個 AI 專案的風險，因此盡早開始了解資料很重要。

您應該至少對於是否能蒐集到足量資料集的難易度有個概念。這很可能會是您面臨到的主要挑戰之一，如果在發現資料根本入手無門之前就投入大量心力，會是個重大災難。

如果現階段還無法進行資料探索，建議您盡快尋找機會開始著手。

資料探索

資料探索也被稱為探索性資料分析（exploratory data analysis, EDA），目標是認識資料集。在這個脈絡下，我們的目標是了解資料集是否有助於解決問題，無論是作為評估演算法效能的方式，還是用作機器學習的訓練資料。

資料探索通常包括以下內容：

統計分析
使用描述性統計來總結資料的特性。

降維
轉換資料，使其更容易分析。

特徵工程
提取有用的訊號，如第 91 頁的「特徵工程」。

視覺化
生成代表資料結構的各種圖表。

建模

訓練機器學習模型以探索資料間的關係。

資料探索是一個廣大又迷人的領域，是資料科學家和機器學習從業者的共同交集。市面上有大量可用於資料探索的軟體，但由於其複雜的概念和術語，對不具備一定程度資料科學背景的使用者來說，可能會覺得有些難以接近。

儘管如此，仍舊有可能在短時間內透過許多相關可用資源來學會一些適當的入門技能[1]。

然而，目前邊緣 AI 面臨的其中一個挑戰是，需要處理的資料大都是以高頻時間序列和高解析度影像的形式出現的感測器資料：這在資料科學領域來說是相對較新的。資料探索工具通常較適用於表格資料、低頻時間序列和文字資料，例如企業財務資料和社交媒體貼文等。這意味著我們很難找到能派上用場的工具和資源。

您可能會發現，在傳統資料科學領域之外具備其他專業知識的工程師多半擁有一些有助於邊緣 AI 資料探索的技能。例如，數位訊號處理的工程師就知道許多善於探索感測器資料的工具，而自然科學家（如生物學家和物理學家）通常在這個領域具備強大的實作技能。

目標設定

我們試著在目標設定中描述出專案的目標。在第六和第八章中都有看到許多關於目標設定的活動。

這個過程包含了以下幾點關鍵要素：

- 確定將用於部署之前與之後的評估指標（第 288 頁「訂下設計目標」）
- 為設計設定系統目標（第 289 頁「系統目標」）

1 Joel Grus 所寫的《Data Science from Scratch: First Principles with Python》是一本專門討論本主題的暢銷書。

回饋循環

圖 9-3 是常見的 AI 開發流程圖，顯示出一個簡單又循序漸進的回饋循環，從資料蒐集開始，最後結束於設備部署。人們很容易陷入這種想法，因為它針對資訊如何在系統中流動提供了一個易於理解的方式。

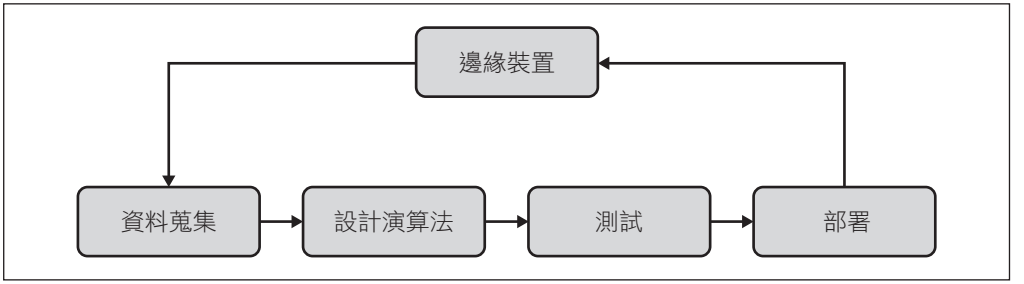

圖 9-3　人們容易將 AI 開發回饋循環視為循序漸進的過程，這是單純把線性工作流程直接變成循環的結果

然而，正如第 177 頁「邊緣 AI 工作流程」中所述，實際上每個系統項目之間都存在著互動。它們彼此之間都有著動態關聯，不容易用基本的圖表就能表達。圖 9-4 更真實地顯示了系統的情況。

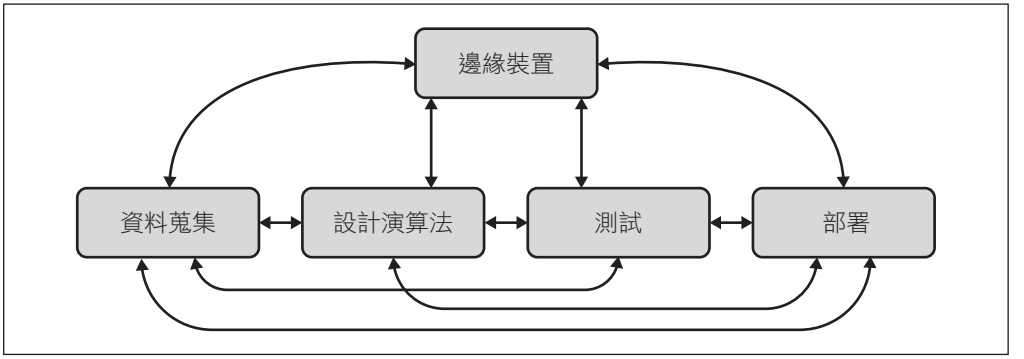

圖 9-4　AI 開發實際上是由多個彼此回饋的項目所組成的網路

在管理專案時，讓回饋可以在任何一點之間自由流動相當重要。例如，資料集的某些方面例如原始資料特定頻段中包含的能量，可能會影響到硬體設計，因為硬體的採樣速度要夠快才能呈現該頻率。反之亦然，如果硬體受限於某些感測器，那麼資料集就必須要能反映出該感測器可捕捉到的訊息。

有些回饋循環比較容易建立。例如，讓負責團隊定期交流便可以建立起資料集和硬體之間的回饋循環。然而，根據各個應用程式，將設備部署到現場並監控這麼做的費用可能相當高昂。也因此出現了各種能以模擬（或類似）方式來「關閉迴圈」的工具，第 351 頁「效能校正」就會深入討論。

以下是開發過程中一些最重要的回饋循環：

演算法和資料集

演算法有著各種不同的資料需求。如果可用的資料豐富，就能夠使用多種不同的演算法。如果可用的資料不多，則能夠順利運作的演算法就比較少。如果需要特定演算法的某種特性，就必須蒐集適當的資料集。

演算法和硬體設計

在綠地專案中，演算法的選擇結果連帶決定了到底有哪些硬體可用，因為只有特定的硬體才能有效執行。而在棕地專案中，演算法的選擇則受限於既有硬體的限制。

演算法和現場效能

所選演算法也將影響現場效能，例如，大型的機器學習模型所能提供的結果也會更好。反之，現場所需的效能也會影響到如何挑選演算法。

資料集和硬體設計

硬體設計通常會影響資料集，因為它將決定哪些感測器可用於蒐集資料。相反地，如果已有特定的資料集可用，其呈現的資料類型或來源也將影響硬體設計。例如，使用型號規格完全相同的感測器會比較好。

資料集和現場表現

如果實際表現不如預期，可能會需要根據系統不足之處來蒐集更多的資料。如果可用的資料有限，可能就會被迫接受實際表現不如預期的這項結果。

反之，如果現場表現受限或出現偏差，這將影響您逐漸蒐集而來的資料以及訓練出來的模型。例如，如果大多數使用產品的人都屬於某個特定族群，可能會往該族群的需求開始過度擬合。

實際進行迭代

迭代的基本概念為，做出某些修改後測量其對目標的影響，並決定下一步該怎麼做。在 AI 開發中，訓練機器學習模型便是典型的迭代。模型訓練中常見的迭代過程如下：

1. 取得資料並將其拆分為訓練、驗證和測試資料集。
2. 以過度擬合為目標，運用訓練資料集訓練一個大型模型[2]。
3. 透過驗證資料集來測量效能。
4. 調整設定以提高驗證效能：添加更多資料，加入正規化或嘗試不同的類型和大小的模型。
5. 再次訓練並測量效能。
6. 一旦模型在驗證資料集上的表現夠好，就改用測試資料集測試。
7. 如果表現不錯，那就太棒啦。如果表現差強人意，則將其丟棄並重新開始。

邊緣 AI 的專案流程也很類似上述做法，但還須涵蓋硬體和應用程式等要素。舉例來說，您透過類似於上述流程做出一個有效的演算法，試著將其部署到所選硬體上並進行實測（例如請潛在使用者測試等）。如果運作順利，太棒了；但如果不順利，就不得不改良。

關鍵在於能否迅速地測試和迭代。如果您在每個迭代上都花很多時間，則校正回歸的代價就更大（改良也許反而更糟，或某些東西不適合，像是模型過大無法適用於現有硬體等），因為您可能已經在一條冤枉路上浪費了許多時間。

如果迭代快速，讓每一次的變更都很小並有辦法馬上測試，則永遠不會浪費太多時間陷入最終與其他系統都不相容的開發泥淖。

若您很幸運地擁有大量的資料集，則模型訓練可能會需要一段時間（數小時、數天甚至數週——雖然邊緣 AI 這類的小模型通常不需要那麼久）。在完成一次長達 48 小時的訓練後卻發現自己在程式碼中犯了一個錯誤而導致模型無效，這可真是惡夢一場啊。

2 透過讓資料發生過度擬合，可以證明模型足以完成代表性建模，也表示整體訓練管線是有效的。

為了縮短每次迭代所需的時間，最好從資料集的子集開始下手。例如，您可以從10%的分層樣本開始（如圖 7-14）。一旦這個子集看似可行，就可以逐漸在後續迭代中加入更多資料來改善模型效能。

善用工具便能夠避開這些問題。例如，專門為邊緣 AI 設計的 AutoML 工具（請見第 157 頁「自動機器學習（AutoML）」）就能考量到各種硬體限制，讓您不必擔心會超出硬體規格。

記得，不只有模型需要迭代：還需要修改並精進從硬體到程式碼的每個部分。為了理解效能的變化，會需要使用正確的指標和評估步驟，稍後在第 331 頁的「評估邊緣 AI 系統」會深入探討。

在設計流程中所訂下的目標（請見第 291 頁「技術目標」）將幫助您了解何時停止迭代，也許是因為無法更接近目標了，又或者是已經超過了它。

迭代式工作流程自然會生成許多產物：資料集、模型、訓練腳本以及所有隨之而來的相依套件。隨時注意這些產物非常重要，否則之後將很難理解並複製出最終結果。正如在第 158 頁「機器學習運營（MLOps）」中所學到的，MLOps 提供了一個可靠的框架來做到這一點。

更新計畫。在執行專案的過程中，您對問題以及處理方法的理解可能會發生相當大的改變。有時候可能是目標明顯變得不切實際、被誤導或與解決核心問題無關等。如果是這樣，不要猶豫，召集利益關係者一起重新評估目標。

話雖如此，目標不應頻繁地更動。相反地，如果需要修正方向，則可以根據現有目標調整專案的需求和規格。

舉例來說，假設您正在設計一款運用影像感測器和人臉辨識來管控進出的智慧門鎖。您的專案目標是讓錯誤接受率趨近於 0%。在開發過程中，您便意識到光是使用視覺技術無法實現這個目標。於是您與利益關係者一起更新了專案範疇，好加入其他感測器來提高系統的可靠度。

在迭代式開發過程中有這樣的發現再正常不過。如果您發現目標需要稍微調整的話，不要慌張——這個過程的目的就是要讓您可以調整方向，以便最終能夠做出一款成功的產品。