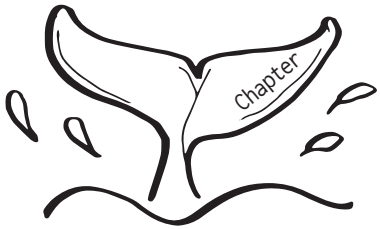


DeepSeek-V3 核心架構及其訓練技術詳解 2

DeepSeek-V3 作為開源的超大規模混合專家（Mixture of Experts, MoE）模型，憑藉創新的架構設計和高效的訓練技術，在效能和資源利用率上實現了突破性進展。

本章深入解析其核心架構，包括混合專家模型的設計原理、動態路由機制和高效參數分配策略，同時探討 FP8 混合精度訓練在降低運算成本和記憶體使用量中的關鍵作用。此外，透過對分散式訓練架構、通訊最佳化技術及負載平衡策略的分析，本章將顯示 DeepSeek-V3 在提升訓練效能和任務適應性方面的技術優勢，為讀者理解現代大模型的訓練方法提供全景視角。

2.1 MoE 架構及其核心概念

MoE 架構是大模型效能提升的重要途徑之一，透過動態路由機制，在每次運算中僅啟動部分專家網路，實現了參數量與運算效能的有機結合。

本節首先介紹 MoE 的基本概念及其在模型擴充中的重要性，其次解析 Sigmoid 路由機制的運作原理及其在動態專家分配中的關鍵作用，最後結合 DeepSeek-V3 的架構設計，顯示其如何利用 MoE 技術在超大規模模型中平衡效能與資源使用量，為高效建模提供技術支撐。

2.1.1 混合專家（MoE）簡介

1. MoE 的基本概念

MoE 架構是一種創新的模型架構，透過加入多個「專家網路」來提升模型的表達能力和運算效能。在 MoE 架構中，多個專家網路被獨立設計為處理不同的特定任務或特定特徵，模型根據輸入資料的特點動態選擇部分專家[1]參與運算，而不是同時啟動所有專家網路。這種「按需運算」的方式大幅減少了資源使用量，同時提升了模型的彈性和任務適用能力。

MoE 的核心概念是透過動態路由機制，在每次推論或訓練中只啟動一部分專家，以在大規模模型中實現參數規模的擴充，而不會顯著增加算力成本。

2. MoE 的優勢與意義

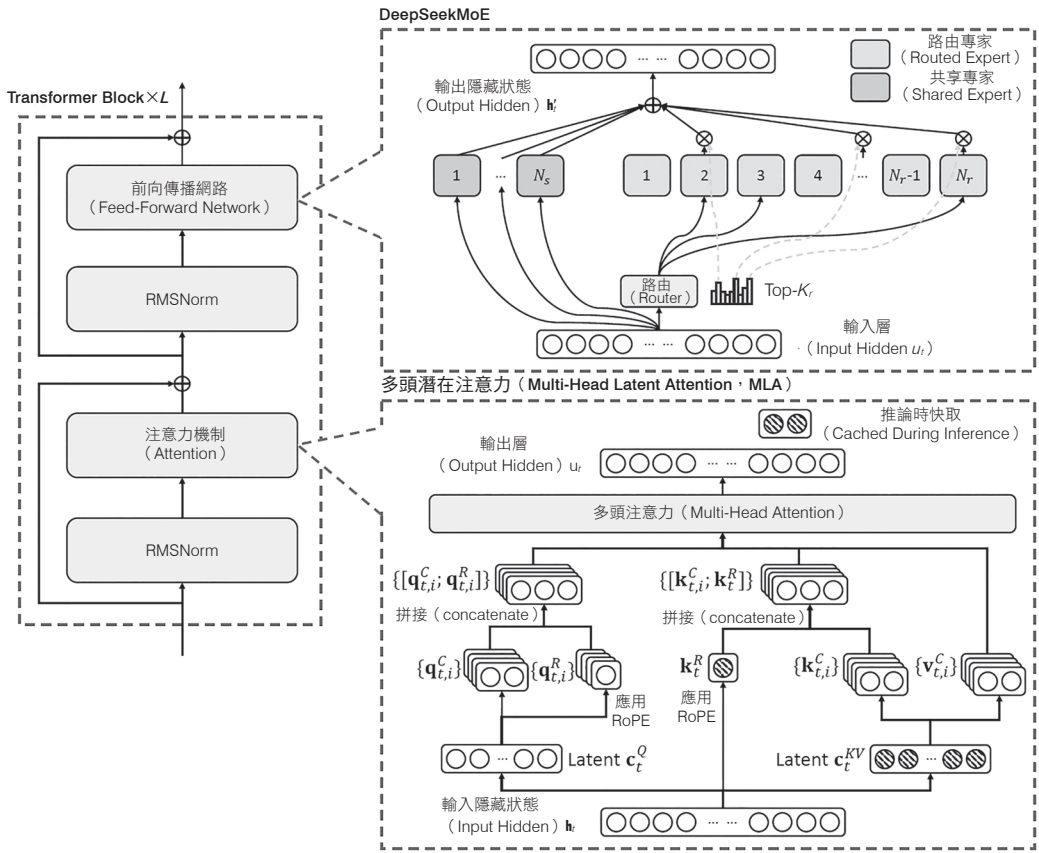
MoE 架構的加入為大規模模型解決了參數擴充與運算效能之間的矛盾，在以下幾個方面形成了優勢。

- 參數規模的擴充：MoE 架構允許模型擁有超大規模的參數量，但每次運算中只需要啟動一小部分參數，就可以大幅提升模型的表達能力。
- 高效資源利用：透過動態選擇專家，MoE 架構避免了運算資源的浪費，同時節省了記憶體和運算成本。
- 任務適用能力增強：不同的專家網路可以針對不同任務進行最佳化，使模型在多任務環境中具備更強的適應性。
- 分散式訓練的相容性：MoE 架構天然適用分散式運算環境，透過將不同的專家網路分布到多個運算節點，顯著提升了並行運算效能。

1 第 2~4 章提到的「專家」均指 MoE 架構中的專家模組。

3. MoE 的任務機制

MoE 架構的關鍵在於其動態路由機制，DeepSeek-V3 中的 MoE 架構及 Transformer 模組的架構如圖 2-1 所示。



▲ 圖 2-1　DeepSeek-V3 整體架構圖 (含 MoE 和 Transformer)

動態路由的主要任務是根據輸入資料的特性，選擇合適的專家網路進行運算，其基本步驟如下：

- 輸入特徵分析：根據輸入資料的特徵，透過路由網路（通常為一個小型神經網路）生成每個專家的啟動機率。
- 專家選擇：根據啟動機率，選取一部分專家網路參與當前輸入的運算。

- 專家運算：被啟動的專家網路對輸入資料進行處理，生成特定的輸出結果。
- 結果聚合：將多個專家網路的輸出結果按照權重進行聚合，生成最終的輸出。

這種按需啟動的機制以確保 MoE 架構能夠在維持高效能的同時，顯著降低了運算量。

4. DeepSeek-V3 中的 MoE 架構應用

DeepSeek-V3 是一個典型的 MoE 架構模型，其創新點主要展現在以下幾個方面。

- 超大規模專家網路：DeepSeek-V3 包含數千個專家網路，每個專家針對特定任務或特定輸入特徵進行了最佳化，以實現了極高的表達能力。
- 動態專家分配：透過高效的路由網路，DeepSeek-V3 能夠根據輸入的特性動態選擇合適的專家，以在不同任務中展現出極高的適應性。
- 高效的稀疏啟動：在每次運算中，DeepSeek-V3 僅啟動少量（如 2~4 個）專家網路，大幅減少了實際運算量和記憶體使用量。
- 分散式訓練最佳化：DeepSeek-V3 將不同的專家網路分布到多個運算節點，透過高效的通訊策略實現了分散式環境下的快速訓練，全過程訓練成本如表 2-1 所示，包括預訓練、擴充訓練及微調訓練等步驟。

▼ 表 2-1　DeepSeek-V3 訓練成本 [2]

訓練成本	預訓練	擴充訓練	微調訓練	總計
H800 GPU 運算時間 / 千小時	2664	119	5	2788
訓練費用 / 美元	5328	238	10	5576

2　相關數據源自 DeepSeek 發布的技術報告。

充分利用了這項技術，在實現大規模模型高效訓練的同時，降低了資源成本，為混合精度運算技術的實務應用建立標竿。

2.2.2 FP8 在大模型訓練中的應用

1. FP8 的基本概念

FP8 是一種新型的低精度浮點數格式，使用 8 位來表示數值，相較於傳統的 32 位浮點數（FP32）或 16 位浮點數（BF16），具有更低的儲存需求和計算複雜度。儘管數值範圍和精度較低，但 FP8 透過結合動態範圍縮放技術和硬體支援，可以在不顯著影響模型效能的情況下，大幅降低記憶體和運算資源的使用量。因此，FP8 成為大模型訓練中的重要工具，為解決記憶體瓶頸、提升運算效能提供了技術支援。

2. FP8 在模型訓練中的核心應用場景

FP8 主要應用於以下模型訓練階段。

- 前向傳播運算：在前向傳播中，權重和啟動值使用 FP8 格式儲存和運算。由於啟動值通常占用較大記憶體，而 FP8 可以減少儲存空間需求，以增加一次性載入的資料量，提升訓練效能。
- 反向傳播梯度運算：反向傳播過程中，大部分梯度運算也可以採用 FP8 精度。FP8 的運算速度更快速速，在硬體支援下能夠大幅提升模型訓練的吞吐量。
- 權重更新中的低精度應用：部分權重更新步驟可以使用 FP8 進行運算，尤其是在動態範圍縮放的配合下，能夠維持數值的穩定性，同時降低運算成本。

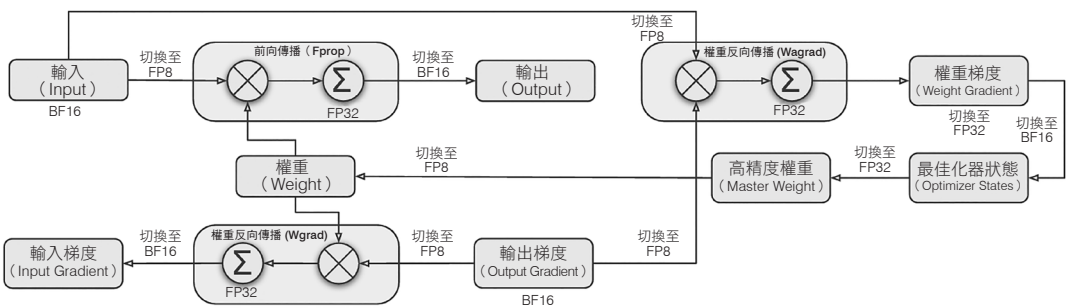
3. FP8 應用所面對的技術挑戰與對應的解決方案

FP8 在實務應用中面對一些技術挑戰，包括數值範圍較小和精度損失問題，但透過以下解決方案，這些問題已獲得有效緩解。

- 動態範圍縮放：FP8 的數值範圍有限，可能使得數值溢位或下溢。動態範圍縮放技術透過調整縮放因子，使數值在合理範圍內分佈，以避免了溢位問題並維持運算穩定性。
- 梯度復原機制：在反向傳播中，如果 FP8 梯度運算的精度不足，模型可以復原到更高精度（如 BF16 或 FP32）進行關鍵梯度的運算，確保權重更新的正確性。
- 硬體最佳化支援：現代硬體（如 NVIDIAHopper 架構）專門為 FP8 設計了運算單元和指令集，顯著提高了 FP8 運算的效能和可靠性，為大規模模型的高效訓練提供了硬體保障。

4. DeepSeek-V3 中 FP8 的具體應用

DeepSeek-V3 是首批全面採用 FP8 混合精度訓練的大規模模型之一，FP8 格式下的混合精度訓練架構如圖 2-2 所示，其在模型訓練中的應用充分顯示了 FP8 的優勢。



▲ 圖 2-2　FP8 格式下的混合精度訓練架構

- FP8 作為主要運算精度：DeepSeek-V3 在前向傳播和梯度運算中廣泛使用 FP8，大幅減少了記憶體使用量，允許更大規模的批次訓練，以提高了硬體使用率。

- 動態精度切換機制：在關鍵運算步驟（如梯度累積和權重更新）中，DeepSeek-V3 結合 FP8 和 BF16 或 FP32 進行動態切換，以確保數值精度與訓練效能的平衡。
- 訓練吞吐量的提升：透過 FP8 的高效運算能力，DeepSeek-V3 在相同的硬體資源下實現了更快速速的訓練速度，為超大規模模型的高效訓練提供了全新的技術解決方案。
- 硬體相容性：DeepSeek-V3 針對支援 FP8 的 GPU 進行了深度最佳化，最大化運用硬體效能，進一步提升了訓練效能。

在實務應用中，FP8 作為低精度運算格式，在降低記憶體使用量、加速模型訓練方面展現了巨大的潛力。透過動態範圍縮放、硬體支援和精度切換等技術手段，FP8 能夠在維持模型效能的同時，顯著降低訓練成本。DeepSeek-V3 全面採用 FP8 技術，在訓練效率與資源利用上取得重大突破，為大規模模型的開發樹立了新的技術標竿。這項技術的應用，不僅推動了混合精度訓練的發展，還為未來大規模模型的高效訓練提供了重要參考。

2.2.3 基於 FP8 的 DeepSeek-V3 效能提升策略

1. FP8 在 DeepSeek-V3 中扮演的核心角色

DeepSeek-V3 全面採用 FP8 作為模型的主要運算精度，透過此低精度格式，實現了在資源使用量與效能表現之間的最佳平衡。FP8 的加入不僅降低了計算複雜度和記憶體使用量，還為模型的快速滾動式調整和部署提供了技術支援。DeepSeek-V3 在 FP8 的基礎上進行了針對性最佳化，透過創新策略進一步提升了訓練和推論的效能。

2. 動態範圍調整技術的應用

FP8 的數值範圍較小，可能使得在訓練過程中出現溢位或下溢問題。為解決此難題，DeepSeek-V3 加入了動態範圍調整技術。

- 逐層範圍最佳化：不同層的啟動值和梯度分布差異較大，動態範圍調整根據每一層的分布特性動態設定縮放因子，確保數值始終處於有效範圍內。
- 自動化調整策略：模型在訓練過程中透過分析數值變化情況，自動決定何時及如何調整精度，在保證模型效能的前提下，提高訓練效能並降低運算資源的使用量。當數值較大或較小時，可動態切換到合適的精度格式，避免因 FP8 數值範圍小而使得的溢位或下溢問題，使訓練過程更加穩定和高效。

3. 混合精度的動態切換機制

儘管 FP8 在多數運算中表現出色，但部分關鍵步驟需要更高的精度支援。DeepSeek-V3 透過混合精度切換策略，在維持 FP8 的高效能的同時解決關鍵運算中的精度問題。

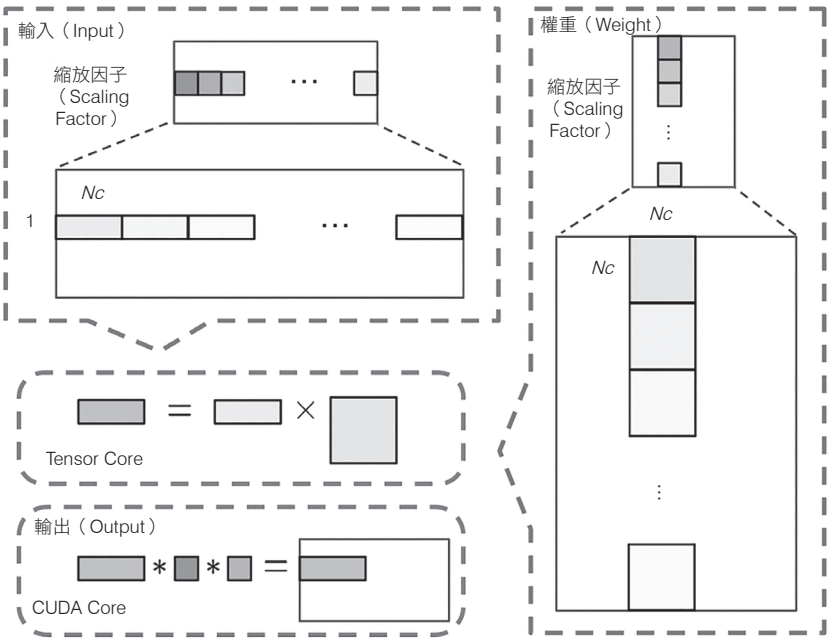
- 梯度累積的高精度切換：在梯度累積階段，模型使用 BF16 或 FP32 儲存和累積梯度，以避免精度損失對權重更新的影響。
- 關鍵權重的高精度儲存：對模型中特定關鍵參數保留更高精度儲存，在推論和訓練中進行低精度讀取和高精度回寫，確保模型效能的穩定性。

4. 高效記憶體管理與批次擴充

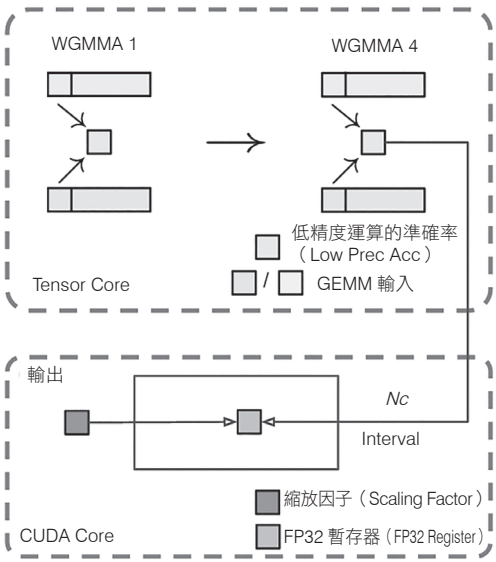
FP8 大幅減少了單次運算的記憶體使用量，DeepSeek-V3 進一步利用此特性最佳化了記憶體管理。

- 批次規模的擴充：FP8 的低儲存需求允許 DeepSeek-V3 在訓練時載入更大的資料批次，以提升訓練吞吐量，減少每個週期的訓練時間。
- 快取與並行最佳化：在記憶體資源有限的環境中，DeepSeek-V3 加入了分散式快取機制和任務並行策略，以充分利用硬體資源。

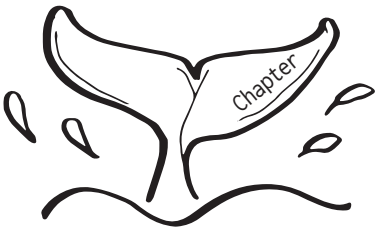
圖 2-3 所示為 DeepSeek-V3 在 FP8 混合精度訓練中的兩項關鍵最佳化技術，包括細粒度量化策略和累加精度提升策略。這些技術的結合有效提升了模型的效能與資源利用效能，尤其在分散式運算中展現出顯著優勢。



▲ 圖 2-3（a）：細粒度量化策略



▲ 圖 2-3（b）：累加精度提升策略



整合實戰 2：AI 助理開發 11

在人工智慧領域，AI 助理（或稱「智慧助理」）是重要的應用，已廣泛應用於日常生活與任務中。本章將深入探討基於 DeepSeek-V3 模型的 AI 助理的開發與實作，透過引入 DeepSeek 開放平台提供的強大 API 介面，結合多模型處理能力，構建一個具備語意理解、任務調度、資訊查詢等多重功能的 AI 助理系統。該系統能夠透過自然語言與使用者進行互動，提供高效、精準的服務，極大地提升任務效率和使用者體驗。

本章內容不僅涵蓋了 AI 助理的核心技術架構，還探討了如何靈活配置和整合不同的模型，以滿足多樣化的業務需求。無論是簡單的對話式查詢，還是複雜的任務管理與決策支援，DeepSeek 模型的優勢都能得到有效發揮，幫助開發者實現高效的智慧助理應用。

11.1 AI 助理：AI 時代的啟動器

在 AI 技術迅速發展的時代，AI 助理已成為提升生產力與使用者體驗的關鍵工具。作為一種整合語音辨識、自然語言處理、機器學習等技術的應用，AI 助理不僅能實現高效率的資訊獲取與任務執行，還能為企業及個人提供智慧化的服務支持。

本節將深入探討 AI 助理的核心功能，解析它如何在多場景下提供個性化與自動化的服務，協助使用者高效完成任務。此外，隨著技術不斷進步，AI 助理正逐漸從傳統的語音助理轉變為更為複雜且多元化的服務平台，在商業化應用中展現出巨大的潛力與前景。從自動化辦公、智慧客服到個性化行銷，AI 助理的應用範圍正快速拓展，成為推動社會各領域智慧化轉型的啟動器。

11.1.1 AI 助理的核心功能解析

AI 助理的核心功能可以歸納為語音識別與自然語言處理、任務管理、資訊檢索、智慧推薦和多模態互動等模組。這些功能使 AI 助理能夠高效地與使用者進行對話，理解並執行複雜任務。例如，借助深度學習模型，AI 助理能夠理解語音或文字輸入，將其轉化為機器可以執行的指令；而在任務管理方面，AI 助理能夠幫助使用者安排日程、提醒待辦事項、執行常規操作等；透過資訊檢索，AI 助理能夠即時從網際網路或資料庫中獲取相關資訊，提供準確的答案或建議。此外，AI 助理的智慧推薦功能能夠根據使用者的行為、偏好和歷史數據提供個人化建議。

在開發基於 DeepSeek-V3 模型的 AI 助理時，核心功能的實現依賴於 DeepSeek 的強大 API 支援，尤其是在自然語言理解、內容生成與上下文管理等方面。

【例 11-1】 透過 DeepSeek 的 API 建置一個可以進行智慧對話的基礎模型。

```
import requests
import json

# API 端點
api_url = "https://api.openai.com/v1/chat/completions"

# 使用者的 API 金鑰
api_key = "your_api_key_here"

# 設定請求標頭
headers = {
```

```
    "Authorization": f"Bearer {api_key}",
    "Content-Type": "application/json"
}

# 定義請求主體，輸入使用者的訊息及聊天上下文
data = {
    "messages": [
        {"role": "system", "content": "你是一個智慧助理，幫助使用者解答問題並提供建議。"},
        {"role": "user", "content": "今天的天氣如何？"}
    ]
}

# 發送 API 請求
response = requests.post(api_url, headers=headers, data=json.dumps(data))

# 檢查回應
if response.status_code == 200:
    result = response.json()
    # 輸出 AI 助理的回覆
    print("AI 助理回覆：", result['choices'][0]['message']['content'])
else:
    print("請求失敗，錯誤訊息：", response.text)
```

案例要點解析：

- API 金鑰：使用者需要在 DeepSeek 平台上註冊並獲取 API 金鑰，用於身分驗證。
- 請求標頭設定：Authorization 欄位使用 Bearer Token 認證。
- 請求主體數據：包含聊天上下文，系統角色設定了智慧助理的基本任務，使用者輸入了「今天的天氣如何？」。
- API 請求：透過 requests.post() 向 DeepSeek API 發送 POST 請求，回傳 AI 助理的聊天回覆。

執行結果：

```
AI 助理回覆：今天的天氣晴朗，氣溫約 28° C，適合外出活動。
```

對以上執行結果解析如下：

- 請求：API 請求透過 POST 方法發送到 DeepSeek 的聊天 API 介面，提供對話上下文及使用者輸入的內容。
- 回應處理：API 回應中包含 AI 生成的訊息，展示 AI 助理的回覆內容。

以上案例展示了 AI 助理在處理文字輸入時，如何透過 DeepSeek 模型的 API 實作自然語言生成與任務處理，根據使用者需求進行適應性對話和任務執行，為 AI 助理功能提供基礎架構支援。

11.1.2 AI 助理的商業化應用

隨著人工智慧技術的不斷進步，AI 助理已經不再局限於實驗室或學術研究領域，它們正逐步滲透到商業化應用中，成為提升企業效率和使用者體驗的核心工具。從智慧客服到個人助理，再到企業級解決方案，AI 助理正在重新定義企業與客戶的互動方式。商業化的 AI 助理不僅能夠提高回應速度和處理能力，還能透過智慧學習與數據分析提供個人化服務。

在商業化應用中，AI 助理主要應用在以下幾個方面：

- 客戶服務：AI 助理被廣泛應用於客戶支援與服務領域，可以處理大量的使用者查詢，並提供 7×24 小時不間斷服務，顯著減輕人工客服的負擔。
- 電商推薦系統：AI 助理可以分析使用者行為、購買歷史等數據，進行個人化推薦，提升銷售轉換率。
- 企業內部效率提升：AI 助理可整合企業內的工作流程和任務管理系統，自動處理日常任務，如行程安排、數據整理等，提升員工生產效率。

借助 DeepSeek-V3 模型的 API，企業可以透過整合 AI 助理技術，在各種應用場景中實現智慧對話與自動化任務，提供更具競爭力的服務。

【例 11-2】 基於 DeepSeekAPI 開發簡單的智慧客服應用。在此應用中，使用者可透過 AI 客服查詢產品資訊並獲取協助。

```
import requests
import json

# DeepSeek API 介面地址
api_url = "https://api.deepseek.com/v3/chat/completion"
api_key = "your_api_key_here"    # 使用者的 API 金鑰

# 設定請求標頭
headers = {
    "Authorization": f"Bearer {api_key}",
    "Content-Type": "application/json"
}

# 定義請求主體，輸入使用者的訊息及聊天上下文
data = {
    "messages": [
        {"role": "system", "content": "你是一位產品客服助理，幫助使用者解答關於產品
的資訊問題。"},
        {"role": "user", "content": "請告訴我你們的最新產品有哪些？"}
    ]
}

# 發送 API 請求
response = requests.post(api_url, headers=headers, data=json.dumps(data))

# 檢查回應結果
if response.status_code == 200:
    result = response.json()
    # 輸出 AI 的回應
    print("AI 客服回應：", result['choices'][0]['message']['content'])
else:
    print("請求失敗，錯誤訊息：", response.text)
```

以下是上述代碼的詳細說明：

- API 金鑰：需要從 DeepSeek 平台獲取 API 金鑰，以進行身分驗證。
- 請求標頭設定：使用 Authorization 欄位，以 Bearer Token 形式傳遞 API 金鑰。

出了更大最佳化，使其更能適應低資源環境；而 DeepSeek-V3 則較注重長文對話中上下文維持的能力，兩者在低資源推論場景中展現出不同的最佳化方向。

綜合來看，DeepSeek-R1 透過低精度運算、稀疏運算、自適應運算以及快取最佳化，使得其在低算力設備上仍能維持穩定高效的推理效能，為資源受限場景提供了更優的解決方案。

A.3 DeepSeek-R1 API 初步開發指南

DeepSeek-R1 提供功能強大的 API 介面，支援高效呼叫推論能力，並適用於多種應用場景。本節將圍繞 API 的基本使用、高級呼叫及效能最佳化進行說明，重點介紹 API 的呼叫方法、請求結構、參數配置與流式輸出方式，同時探討函式呼叫、JSON 模式及多次對話等高級功能，並與 DeepSeek-V3 的 API 進行對比，解析兩者在互動方式、推論效能及適用場景上的差異，為建置高效智慧應用提供技術指引。

API 呼叫與基本使用

DeepSeek-R1 的 API 與 OpenAI 的 API 相容，因此可採用 OpenAI 的 SDK 進行呼叫。以下為使用 Python 呼叫 DeepSeek-R1 API 的範例程式碼：

```
import requests
import json

# 設定 API 請求的 URL
url = "https://api.deepseek.com/v1/chat/completions"

# 建置請求的有效負載
payload = {
    "model": "deepseek-reasoner",
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "請介紹一下 DeepSeek-R1 的主要特點。"}
    ]
}
```

```
# 設定請求標頭，包括授權資訊
headers = {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer YOUR_API_KEY'  # 將 YOUR_API_KEY 取代為實際的 API 金鑰
}

# 傳送到 POST 請求
response = requests.post(url, headers=headers, data=json.dumps(payload))

# 輸出回應結果
if response.status_code == 200:
    result = response.json()
    print(result['choices'][0]['message']['content'])
else:
    print(f"請求失敗，狀態碼：{response.status_code}")
```

上述程式碼中，首先匯入 requests 與 json 模組，接著設定 API 請求 URL。在 payload 中指定模型為 "deepseek-reasoner"（即 DeepSeek-R1 模型），並於 messages 列表中提供對話上下文，其中 role 可設定為 "system"（系統）、"user"（使用者）或 "assistant"（助手），content 則包含具體訊息內容。

在 headers 中，設定 Content-Type 為 application/json，並在 Authorization 欄位加入 Bearer token，後接使用者的 API 金鑰。接著利用 requests.post 方法傳送到 POST 請求，並將 payload 轉為 JSON 格式。最後，根據回應狀態碼（200 表示成功），解析 JSON 回應並輸出助手的回答內容。

讀者需注意，DeepSeek-R1 的 API 具備速率限制與費用運算，因此在開發過程中應遵守 API 使用規範，避免頻繁請求使得限流。此外，API 呼叫會依據輸入與輸出之 token 數量計費，設計應用時請考慮最佳化請求內容以降低使用成本。

透過上述步驟，讀者即可在 Python 環境中成功呼叫 DeepSeek-R1 的 API，與模型進行互動，並依需求調整請求參數以充分發揮 DeepSeek-R1 的推論能力。