AAS

**ABSTRACT:**12313

**Keywords:** 45646

#####主體架構參考paper#####

# 參考文獻

# Introduction

近年來，神經網路驅動的大型語言模型（Large Language Models, LLMs）應用於軟體開發的情形顯著增加，其中包括GPT-4、Cloude、Genmini等模型。這些模型在從自然語言提示自動產生程式碼的任務中已被證實具有成功性(Chen et al., 2024)。LLM能夠從直觀的自然語言輸入中產生程式碼，為開發者提供了有前景的應用途徑，有望簡化程式設計流程並加速軟體開發週期(Haque, 2025)。在教育領域中，LLM能夠產生自然語言文本，並提供即時且具情境關聯的回應，可應用於寫作輔助、補充教材、程式語言學習指引，以及教師進行評量與教材生成的支援(Steiss et al., 2024)。

最初LLM工具主要用於文本生成，但隨著自然語言處理的進步，LLM已逐步擴展至更具互動性的角色，例如提供個人化學習體驗、回饋機制、專科輔導，甚至能生成語音、影像、影片等多種媒體形式(Bozkurt, 2023)。將LLM納入教育情境具有多重優勢。第一，LLM提供隨需支援，讓學生在任何時候都能獲得幫助，特別適合自我節奏學習的情境(Cao & Dede, 2023)。第二，LLM可以因應不同的學習風格，透過多種格式（如文字說明、摘要、逐步引導）提供教學說明，並已證實能提升學習成效與參與度(Pesovski et al., 2024)。第三，LLM可協助教師自動化處理重複性工作，例如批改評量與設計教案，使教師能將更多時間投入在個別化教學與學生互動上(Khazanchi & Khazanchi, 2024)。

儘管優點顯著，LLM在教育應用中仍面臨若干挑戰與限制。最主要的疑慮在於生成內容的品質與正確性。由於LLM是在龐大資料集上訓練而成，資料集本身同時涵蓋正確資訊與錯誤資訊，因此有時會生成不準確或具偏見的回應(Doughty et al., 2024)。如果模型缺乏深層語境理解，往往會往產生空泛或無關的資訊(Weisz et al., 2024)。為了處理上述挑戰，一種解法是使用優質資料與人為回饋來微調LLM，以提高其準確性與情境對應能力(Dave et al., 2023)。另一種方法是透過提示工程（Prompt Engineering），透過Prompt設計，給予精確且具情境性的輸入引導模型回應(Alier et al., 2024)。零樣本（zero-shot）與少樣本（few-shot）提示法可以讓模型在極少的訓練資料下執行新任務，善用既有知識產出準確答案(Brown et al., 2020)。Chain-of-Thought（CoT）提示法透過將任務拆解為邏輯步驟來提升模型推理能力，在處理複雜問題時能增進準確性(Zhang et al., 2022)。

在教育脈絡中，提示訓練（prompt training）是一種教學介入策略，可幫助學生更有效地利用 AI 工具，進而提升學習表現與學科理解深度(Walter, 2024)。然而，目前針對提示技能與學生學習表現間關聯的研究仍十分有限，顯示此領域仍有待深入探討。

WWWWWWWWWWWWWWWWWWWWWWWWWW

# 研究問題：

RQ1：使用Chain-of-Thought（CoT）提示優化，對於 LLM 所生成 Python 程式碼的執行時間、記憶體使用量與峰值記憶體使用量影響為何？

RQ2：對 LLM 進行特定資料集微調（fine-tuning），是否能改善其所生成程式碼在執行時間、記憶體使用量與峰值記憶體使用量上的表現？

# Related Work01：評估程式碼

在教育場域中，學生使用大型語言模型（LLM）協助撰寫程式碼已日益普遍。

然而，這些先進神經網路工具的預測性質也引發了跨領域的可信度疑慮，例如自然語言處理(Deanda et al., 2015)、電腦視覺(Zhang et al., 2019)、自動駕駛(He et al., 2024)與醫療保健領域(Deanda et al., 2025)。程式碼生成的脈絡中，主要關注於LLM所生成程式碼的實際可用性，尤其在硬體受限下的環境更需重視效能指標 (Coignion et al., 2024; Qiu et al., 2024)。

程式碼效能的評估指標通常包括記憶體使用量、CPU使用率、執行時間與程式碼複雜度。Dou et al., (2024)與Hou and Ji (2025)比較了LLM與人類程式設計師的表現。發現GPT-4 雖然在結構化任務中表現優異，但在面對模糊任務時表現不佳，而人類設計師撰寫的程式碼在執行時間與記憶體使用方面通常更有效率，突顯出 LLM 與人類解法在優化方面的落差。在Niu et al., (2024)的研究中，除了正確性外，研究也納入執行時間作為效率指標，並在 HumanEval、MBPP 這些基準資料集中(聚焦 Python 問題)，與LeetCodeEval資料集（涵蓋 C++ 題目）上進行測試。結果指出，LLM的執行效率仍落後於人類所撰寫的程式碼。

為了評估生成程式碼的正確性，即是否達成預期功能。Paul et al.,(2024)分析了現有的基準資料集。例如HUMANEVAL(Chen et al., 2021)、MBPP(Austin et al., 2021)等方式來評估LLM所產出的程式碼。相關研究採用pass@k作為效能評估指標。pass@k意旨在k次生成中至少成功一次的機率。然而Paul et al.,(2024)所研究的重點為 LLM 所產生程式碼的正確性，但對效率問題則未多加關注。Du et al., (2024)則提到，程式碼的效率應該被定義在資源消耗最小的情況下完成任務的能力。這樣的程式碼才能提升使用者體驗、節省能源，並使應用更具成本效益。

儘管近期的基準研究開始重視程式碼的功能正確性，像是EFFIBENCH是一套專門用來評估LLM生成程式碼正確性與效率的測試集(Huang et al., 2024)。但研究也指出，即使LLM生成的程式碼在功能上是正確的，卻經常在記憶體使用與執行時間方面不如人類撰寫的程式碼(Huang et al., 2024)。此結果源自目前主流的訓練範式，即「下一個程式碼 token 預測」傾向於強調局部模式的補全，而非尋找整體上更有效率的演算法結構(Guo et al., 2024)。儘管語法上正確，但局部補全的方法會產生多餘的程式碼區塊，導致時間與空間複雜度上升，進而造成計算資源的浪費。LLM生成程式碼的固有低效率，會使應用程式的執行速度變慢、運作成本上升，甚至無法符合系統需求。尤其是IoT、邊緣運算系統、雲端裝置等資源受限的環境下部署時，記憶體使用量與執行速度為關鍵考量(Bolón-Canedo et al., 2024; Solovyeva et al., 2025)。

在訓練資料集上，同時也存在有效與無效的程式碼，這會讓模型對於提示語意產生幻覺(hallucination)，進一步導致不必要的能源消耗與在關鍵應用場景下的效能下降(Huang et al., 2025)。Ehsani et al., (2025)與Khojah et al., (2025)也指出，若prompt本身含糊或結構不良，模型可能誤解任務本意，進而產出不必要複雜或效率低落的解決方案。

# Related Work02：提示工程（Prompt Engineering）

提示工程（Prompt Engineering）的發展歷程始於「零樣本提示（Zero-Shot Prompting）」，這種方法讓模型在沒有特定訓練資料的情況下，僅依據一般性的指令執行任務。隨後「少樣本提示（Few-Shot Prompting）」的出現引入少量範例，以協助模型更準確地理解任務內容(Brown et al., 2020)。Chain-of-Thought（CoT）由Wei et al., (2022) 提出，進一步推進提示技術，透過逐步推理的方式，顯著提升模型在推理任務中的表現。Niu et al., (2024) 採用了不同的提示設計技巧，例如針對簡單問題使用簡易提示（simple prompt），針對複雜問題則使用Chain-of-Thought（CoT）提示。CoT將問題分解為邏輯步驟，藉此減少冗餘、提升效率。結果顯示，CoT 可使執行時間降低 15%，記憶體使用量降低 12%，是一種實用的優化工具(Wei et al., 2022)。

在自動化流程上，Zhang et al., (2022) 提出了「自動化 Chain-of-Thought（Auto-CoT）提示」，能自動產生多樣化的推理鏈，在無需人工設計的情況下提升準確度與穩定性。Wang et al., (2022) 在 CoT 提示的基礎上提出「自我一致性（Self-Consistency）」機制，透過產生多條推理路徑，並選出最一致的答案，藉此進一步提升解題準確性與穩健性。Zhao et al., (2023) 提出了「邏輯推理提示（LogiCoT）」，將符號邏輯原則引入推理過程，用於驗證每個推理步驟的正確性，以降低錯誤與幻覺（hallucination）的發生。

而為了有效實作上述提示技術，研究者也設計了多種提示框架，使這些方法得以實際應用於現實世界中。提示框架在銜接模型能力與使用者實際需求之間，扮演了關鍵橋樑的角色。這些框架提供必要的基礎設施，如整合外部工具、維持歷史資訊、確保輸出結果具備結構性與安全性(Liu et al., 2023)。例如LangChain 與 Semantic Kernel 等框架可讓 LLM 與資料庫、網頁瀏覽器及其他外部系統互動，進而克服 LLM 的內在限制並擴展其應用範疇(Liu et al., 2023)。透過一個全面而可調適的結構，全面提升人機溝通的品質（Lo, 2023）。