期末專題可行性評估：基於樹莓派 4 之影像辨識與測溫系統

I. 執行摘要：高階可行性與策略性建議

本報告旨在評估一項期末專題的可行性，該專題涉及使用樹莓派 4 (4GB) 進行影像辨識與測溫任務。經深入分析，此專案在所提出的兩種架構下皆具有技術可行性。然而，最終的成功與否將取決於對效能、複雜性與資源限制的權衡。這兩種架構分別為：所有任務均在樹莓派上獨立完成的「獨立式架構」，以及將運算任務分流至 Mac 進行分析的「分散式架構」。

對於時間有限的學術或業餘專案，強烈建議採用分散式架構。此方法將最耗費運算資源的任務（影像辨識與分析）轉移至 Mac 上運算能力更強大的硬體，包括其圖形處理單元 (GPU)，從而將樹莓派的角色簡化為一個高效且穩定的資料擷取節點。

此策略規避了獨立式模型中最具挑戰性的技術障礙，例如在資源受限的單板電腦上，要讓複雜的電腦視覺演算法達到即時效能，需要進行廣泛的底層軟體最佳化。分散式模型將專案重點從繁瑣的底層效能調校，轉移至更具可控性與可預測性的高階系統整合，為專案的成功展示提供了一條更為順暢的途徑。

II. 可行性分析：獨立式樹莓派 4

2.1 硬體與軟體基礎：樹莓派 4 作為邊緣運算裝置

樹莓派 4 (4GB) 搭載四核心 ARM Cortex-A72 處理器，使其成為一個功能多樣的 Linux 開發板，並廣泛應用於「邊緣人工智慧」（Edge AI）專案中 [1]。其 4GB 的記憶體是顯著優勢，為資料緩衝區和較大型的記憶體內模型提供了充足空間，這是低階樹莓派型號所不具備的。此處理器基於 ARMv8 架構，支援 NEON 和浮點單元 (FPU) 等關鍵指令集，這些指令對於電腦視覺任務中的向

量化和浮點運算至關重要 [3]。

然而，一個關鍵的技術限制是，樹莓派 4 缺乏專用的硬體加速器。相較之下，較新的樹莓派 5 配備了 Hailo-8L AI 加速器 [5]，而樹莓派 AI 攝影機 (Raspberry Pi AI Camera) 則內建 Sony IMX500 智慧視覺感應器，可為視覺 AI 應用程式提供高效能 [6]。這意味著，樹莓派 4 獨立完成專案的挑戰，並非在於其是否能執行任務，而是在於能否僅透過其通用型 CPU 和 GPU 的軟體最佳化，達到可接受的即時效能。這項硬體差異為專案的範圍和複雜性設定了明確的界線。

2.2 邊緣電腦視覺：效能與最佳化

在樹莓派 4 上進行影像辨識是可行的。研究表明，使用如 Haar Cascades 和定向梯度直方圖 (HOG) + 線性支持向量機 (SVM) 等傳統演算法，可以實現即時人臉偵測 [7]。然而，要達到最佳效能，強烈依賴於從原始碼編譯 OpenCV 函式庫。這樣做可以啟用硬體專用指令集，例如

-DENABLE_NEON=ON，從而使程式碼針對 ARM 晶片進行優化，最高可帶來 **30%** 的速度提升 [4]。

值得注意的是，這個編譯過程本身就是一項耗時且資源密集型的任務。為避免系統在編譯期間因記憶體不足而掛起，必須暫時將系統的交換空間（swap space）擴大至至少 2GB [3]。這項作業不僅是簡單的安裝步驟，而是專案成功的基礎里程碑。如果未能執行這項複雜的底層最佳化，專案的視覺組件可能因運算緩慢而無法滿足即時應用的要求。此外，對於更先進的任務，如部署深度學習模型，效能量測顯示，經過最佳化後，SqueezeNet 的推論時間約為 0.9 秒 [9]。另一項重要的效能最佳化技術是使用

int8 量化模型，這對於在 ARM 處理器上部署 TensorFlow Lite (TFLite) 模型特別有效 [3]。

下表提供了在樹莓派上執行不同電腦視覺演算法時，預期的效能基準，藉此量化最佳化帶來的實際效益：

表 1：樹莓派獨立式效能基準

演算法/模型	預估 FPS (未最佳化)	預估 FPS (已最佳化)	推論延遲	記憶體用量
Haar Cascades	15-20 FPS	20-30+ FPS	低	低
HOG + SVM	1-5 FPS	2-8 FPS	中等	中等

輕量級 DNN (如 MobileNet)	0.5-1 FPS	0.7-1.5 FPS	高	高

2.3 測溫與資料擷取

測溫任務的硬體整合相對簡單。Adafruit MLX90640 紅外線熱像儀是一個廣為人知的元件，用於此類專案 [10]。該裝置透過 I²C 介面與樹莓派通訊，使用者需在作業系統層級使用

raspi-config 工具來啟用此介面 [13]。無論是使用 Python 的

adafruit-circuitpython-mlx90640 函式庫 [15] 或 Go 的

d2r2/go-i2c 函式庫 [13]，都可輕鬆實現軟體層的資料讀取。此硬體的整合挑戰主要在於如何將從 I²C 讀取的熱感資料，與來自攝影機串流介面 (CSI) 的影像資料，在軟體層進行同步。

2.4 感測器融合與並行運算的挑戰

專案的核心挑戰之一是將來自兩個不同感測器（可見光攝影機和熱像儀）的資料進行實時同步與融合 [16]。現有專案已證明，透過 Python 腳本和 OpenCV 視窗，可以成功地將這兩種資料流融合成單一的視覺輸出 [10]。

傳統上，這種同步問題會透過多執行緒和互斥鎖（mutexes）來解決，但這會引入效能開銷，並可能因資源競爭而導致瓶頸 [18]。此時，Go 語言的並行模型提供了一個更優雅的解決方案。Go 的

goroutines (協程) 是極為輕量級的執行緒，而 channels (通道) 則提供了一種安全、高效的方式，讓這些協程之間能夠傳遞資料，避免了共享記憶體所帶來的資料競爭問題 [18]。

channel bridging 是一種特別適用於此專案的設計模式 [19]。它能將來自多個併發來源（例如，一個讀取熱感資料的

goroutine 和一個處理影像資料的 goroutine）的資料流，匯聚到一個單一的通道中進行後續處理，從而簡化了程式碼邏輯，同時保持了系統的並行性。這種方法突顯出 Go 語言在處理即時、資源受限的嵌入式應用中，所具備的根本性優勢。

III. 可行性分析：分散式樹莓派 + Mac 架構

3.1 架構設計與資料流

分散式架構將樹莓派作為專門的資料擷取節點，並將運算任務轉移至 Mac 主機。這類設計與現有的「瀏覽器串流機器人」專案相似，其中樹莓派負責捕捉影像，並透過網路傳輸至另一台電腦進行控制與顯示 [20]。資料通訊可以採用簡單的 HTTP 伺服器，或為了實現低延遲同步，可選擇更為複雜的 Socket (通訊端) 通訊協定 [21]。

3.2 效能與延遲分析

這種架構的效能權衡點從樹莓派的運算延遲，轉移至網路延遲與資料序列化的開銷。儘管 Mac 的運算能力遠超樹莓派，但整個系統的端到端（end-to-end）延遲將受限於無線網路的速度和可靠性。這項分析顯示，主要瓶頸從樹莓派的 CPU 轉移到了網路介面。例如，如果樹莓派傳輸未經壓縮的、高解析度的影像串流，將會迅速佔用網路頻寬，抵銷 Mac 運算能力的優勢。因此，若要確保系統的整體效能，必須專注於高效的資料壓縮與傳輸協定。現有的研究證實，在區域網路環境下，透過適當的串流處理，這種架構的延遲是可控的 [20]。

表 2：架構比較矩陣

評估標準	獨立式樹莓派	分散式樹莓派 + Mac
總體成本	較低（僅樹莓派與感測器）	較高（額外需要 Mac）
系統複雜性	高（需進行複雜的軟體最佳化與並行處理）	中等（重點在於網路通訊與系統整合）
運算延遲	受限於樹莓派的處理器效能	幾乎不受 Mac 處理器限制，取決於網路延遲
可擴展性	低（難以擴展運算能力）	高（可輕易升級 Mac 或使用多台主機）

專案可靠性	較低（軟體環境設定複雜，易出錯）	較高（任務分工明確，故障排除較易）

3.3 軟體堆疊與互通性

Mac 端的軟體設定需要特定的步驟來啟用 GPU 加速。搭載 Apple Silicon 或 AMD GPU 的 Mac 電腦，可以利用 tensorflow-metal 外掛程式來加速 TensorFlow 模型的訓練與推論 [22]。標準的 TensorFlow

pip 套件目前在 macOS 上缺乏官方 GPU 支援，因此此一外掛程式至關重要 [23]。

開放神經網路交換格式（ONNX）在此架構中扮演著關鍵角色。ONNX 是一種開放標準，用於機器學習模型的互通性，允許在一個框架（例如在 Mac 上用 TensorFlow 訓練）中開發的模型，能夠在另一個執行環境（例如在樹莓派上用 ONNX Runtime 執行）中進行推論 [24]。

onnxruntime_go 函式庫為 Go 提供了 ONNX Runtime 的包裝，且其預編譯的二進位檔同時支援 ARM64 Darwin（Mac）和 ARM64 Linux（樹莓派）平台 [25]。這項特性提供了一個強大且跨平台的解決方案，使得模型的部署變得無縫接軌，同時也驗證了在兩個不同平台上使用不同語言進行開發的戰略可行性。

IV. 語言選擇的細緻比較：Go 與 Python

4.1 Python 在嵌入式機器學習中的生態系統

Python 是人工智慧與機器學習領域的「事實標準」語言，這主要歸功於其龐大且成熟的函式庫生態系統，包括 TensorFlow、PyTorch 和 OpenCV [26]。它因其快速的開發週期和作為高性能 C/C++ 函式庫的「膠水」語言而受到讚譽 [27]。對於專案中的 Mac 分析部分，Python 無疑是最容易上手的選擇，能夠最快地建立功能原型。

然而，在處理即時、運算密集型任務時，Python 的解釋型本質和全域解釋器鎖（GIL）可能會導致執行時效能瓶頸 [26]。這意味著，雖然 Python 能夠快速驗證專案概念，但它可能不是最終部署在資源受限的樹莓派上，以追求最佳效能的理想選擇。因此，在開發速度與最終的系統效能之間，存

在一個根本性的權衡。

4.2 Go 在並行、嵌入式系統中的生態系統

Go 是一種編譯型語言，具有「快速執行、低記憶體佔用」的特性，並內建了強大的並行模型 [27]。這使其成為資源受限系統和即時資料處理的絕佳選擇 [29]。儘管其機器學習生態系統相較於 Python 仍不夠成熟 [26]，但用於核心任務的函式庫已然存在，例如 I²C (

d2r2/go-i2c) [13] 和電腦視覺 (

GoCV) [30]。

此專案的語言選擇並非一個非此即彼的決策。最有效的策略是採用混合方法：在 Mac 上使用 Python 進行原型開發和模型訓練，然後在樹莓派上使用效能導向的 Go 語言，建立一個穩健、高效的資料管道。Go 語言的並行性與編譯效能使其非常適合處理樹莓派端的低層級、高性能資料擷取任務。這種策略性地結合兩種語言的優勢，可以讓專案同時享有快速開發與最終效能的益處。

表 3：**Go** 與 **Python** 在嵌入式人工智慧**/**機器學習中的比較

評估標準	Python (分析)	Go (分析)
執行時效能	較慢（解釋型，有 GIL）	較快（編譯型，並行模型原生）
函式庫生態系統	極其豐富，為 AI/ML 首選	較不成熟，部分功能需自行實現
並行處理支援	透過多進程或專用函式庫實現，有 GIL 限制	透過 goroutines 和 channels 原生支援，高效輕量
開發速度	極快，適合快速原型開發	較快，但因嚴格的靜態型別需更多前期工作
記憶體佔用	較大（動態型別、虛擬環境）	較小（編譯型、優化的記憶體管理）

V. 可行的建議與實施路線圖

5.1 最終架構建議

基於上述分析，強烈建議採用分散式樹莓派 **+ Mac** 架構。此模型允許專案核心的運算密集型任務受益於 Mac 的強大處理器和 GPU 加速，同時將樹莓派的複雜性降至最低。這將大幅提高專案成功的可靠性，並讓使用者能將精力集中在更高層次的軟體整合與應用邏輯上，而非底層的效能調校。

5.2 物料清單與軟體依賴

硬體：

- 樹莓派 4 (4GB)
- 樹莓派官方相機模組 3 (Raspberry Pi Camera Module 3) [10]
- MLX90640 紅外線熱像儀 breakout [11]
- Mac 電腦（建議為搭載 Apple Silicon 的型號）
- SD 卡 (建議 32GB 或以上)
- 傳輸線材與電源供應器

軟體（樹莓派端）：

- Raspberry Pi OS (64-bit)
- Go 1.22 或更高版本
- go get github.com/d2r2/go-i2c
- go get github.com/hybridgroup/gocv
- go get github.com/yalue/onnxruntime_go
- i2c-tools 套件

軟體（**Mac** 端）：

- macOS 12.0 (Monterey) 或更高版本 [22]
- Python 3.9 或更高版本 [22]
- tensorflow-metal 外掛程式：python -m pip install tensorflow-metal [22]

- ONNX Runtime Go 函式庫：go get github.com/yalue/onnxruntime_go
- 其他必要的 Python 函式庫（如 NumPy、Pandas）

5.3 逐步實施指南

1. 硬體與作業系統設置：
   - 將 Raspberry Pi OS 燒錄到 SD 卡，並將其插入樹莓派。
   - 使用 raspi-config 工具啟用 SSH 和 I²C 介面 [13]。
   - 連接相機模組與 MLX90640 感測器。
2. 樹莓派資料擷取管道：
   - 在樹莓派上安裝 Go 語言環境。
   - 開發一個 Go 程式，使用 go-i2c 函式庫從 MLX90640 讀取熱感資料。
   - 另一個 goroutine 則負責從相機擷取影像資料。
   - 利用 Go 的 channels 實現兩個資料流的同步與融合。
3. 網路通訊：
   - 在樹莓派上建立一個輕量級的網路服務（例如 gRPC 服務），負責將同步後的資料流傳輸到 Mac。
   - 該服務將熱感資料和影像資料打包，並進行高效的序列化與壓縮，以最小化網路延遲。
4. **Mac** 分析後端：
   - 在 Mac 上安裝 Python 環境，並創建一個虛擬環境（venv）以管理依賴 [22]。
   - 安裝 TensorFlow 與 tensorflow-metal 外掛程式，以啟用 GPU 加速。
   - 開發一個 Python 腳本，作為 gRPC 客戶端，接收來自樹莓派的資料流。
   - 此腳本將接收到的影像資料輸入到預先訓練好的 ONNX 格式模型中進行影像辨識。
5. 最終展示與視覺化：
   - 建立一個圖形使用者介面（GUI）或網頁介面，在 Mac 上實時顯示來自樹莓派的熱感影像、可見光影像，以及影像辨識的結果。這將提供一個直觀、互動的專案成果展示。

引用的著作

1. Raspberry Pi 4 - Edge Impulse Documentation, 檢索日期：9月 20, 2025, https://docs.edgeimpulse.com/hardware/boards/raspberry-pi-4
2. Raspberry Pi with OpenCV: Getting Hands-On with AI at the Edge, 檢索日期：9月 20, 2025, https://opencv.org/blog/raspberry-pi-with-opencv/
3. Install OpenCV on Raspberry Pi 5 - Q-engineering, 檢索日期：9月 20, 2025, https://qengineering.eu/install%20opencv%20on%20raspberry%20pi%205.html
4. Build and Install OpenCV 4 for Raspberry Pi - LearnOpenCV, 檢索日期：9月 20, 2025, https://learnopencv.com/build-and-install-opencv-4-for-raspberry-pi/
5. Buy a Raspberry Pi AI Kit, 檢索日期：9月 20, 2025, https://www.raspberrypi.com/products/ai-kit/
6. Buy a Raspberry Pi, 檢索日期：9月 20, 2025, https://www.raspberrypi.com/products/

7. Real-time face detection on a Raspberry PI 13 (2) 2022 - Problems of Information Society, 検索日期：9月 20, 2025, https://jpis.az/uploads/article/en/2022_2/REAL-TIME_FACE_DETECTION_ON_A_RASPBERRY_PI.pdf
8. [Question] OpenCV performance on Raspberry Pi 4 4GB - Reddit, 検索日期：9月 20, 2025, https://www.reddit.com/r/opencv/comments/gpqq56/question_opencv_performance_on_raspberry_pi_4_4gb/
9. Optimizing OpenCV on the Raspberry Pi - PyImageSearch, 検索日期：9月 20, 2025, https://pyimagesearch.com/2017/10/09/optimizing-opencv-on-the-raspberry-pi/
10. Raspberry Pi Thermal Camera - YouTube, 検索日期：9月 20, 2025, https://www.youtube.com/watch?v=rG0JHzKA_fA
11. Making a thermal camera with a Raspberry Pi just got a ton easier - XDA Developers, 検索日期：9月 20, 2025, https://www.xda-developers.com/thermal-camera-raspberry-pi/
12. Raspberry Pi Thermal Camera - Adafruit Learning System, 検索日期：9月 20, 2025 , https://learn.adafruit.com/raspberry-pi-thermal-camera?view=all
13. d2r2/go-i2c: Implementation of I2C-bus written in Golang. Forked from davecheney/i2c., 検索日期：9月 20, 2025, https://github.com/d2r2/go-i2c
14. Raspberry Pi I2C (Python) : 7 Steps (with Pictures) - Instructables, 検索日期：9月 20, 2025, https://www.instructables.com/Raspberry-Pi-I2C-Python/
15. adafruit/Adafruit_CircuitPython_MLX90640: A pure Python MLX90640 driver - GitHub, 検索日期：9月 20, 2025, https://github.com/adafruit/Adafruit_CircuitPython_MLX90640
16. Sensor fusion - Wikipedia, 検索日期：9月 20, 2025, https://en.wikipedia.org/wiki/Sensor_fusion
17. Sensor Fusion Explained: The Future of Embedded Systems - DISTek Integration, 検索日期：9月 20, 2025, https://distek.com/blog/sensor-fusion-explained-the-future-of-embedded-systems/
18. Golang Concurrency: How Confinement Improves Performance Without Locks, 検索日期：9月 20, 2025, https://dev.to/bahaanoah/golang-concurrency-how-confinement-improves-performance-without-locks-1ld5
19. Mastering Go Concurrency: A Comprehensive Guide to Channel Bridging | by Sourav Choudhary | Medium, 検索日期：9月 20, 2025, https://medium.com/@souravchoudhary0306/mastering-go-concurrency-a-comprehensive-guide-to-channel-bridging-3b612df453dd
20. Build a Video Streaming Robot with the GoPiGo3 Raspberry Pi Robot - Dexter Industries, 検索日期：9月 20, 2025, https://www.dexterindustries.com/GoPiGo/projects/python-examples-for-the-raspberry-pi/browser-video-streaming-robot-gopigo3/
21. How to synchronize multiple video displays? - Raspberry Pi Forums, 検索日期：9月 20, 2025, https://forums.raspberrypi.com/viewtopic.php?t=113498

22. Tensorflow Plugin - Metal - Apple Developer, 検索日期：9月 20, 2025, https://developer.apple.com/metal/tensorflow-plugin/
23. Install TensorFlow with pip, 検索日期：9月 20, 2025, https://www.tensorflow.org/install/pip
24. ONNX | Home, 検索日期：9月 20, 2025, https://onnx.ai/
25. yalue/onnxruntime_go: A Go (golang) library wrapping microsoft/onnxruntime. - GitHub, 検索日期：9月 20, 2025, https://github.com/yalue/onnxruntime_go
26. Golang vs Python for AI Solutions: How Do You Decide? - Reddit, 検索日期：9月 20, 2025, https://www.reddit.com/r/golang/comments/1hcjh6i/golang_vs_python_for_ai_solutions_how_do_you/
27. Go vs. Python, which is better for AI? - Quora, 検索日期：9月 20, 2025, https://www.quora.com/Go-vs-Python-which-is-better-for-AI
28. Go vs Python vs Rust: Which One Should You Learn in 2025? Benchmarks, Jobs & Trade-offs - PullFlow, 検索日期：9月 20, 2025, https://pullflow.com/blog/go-vs-python-vs-rust-complete-performance-comparison
29. Go vs Python: Pick the Language for Your Project | Guide 2025 - Mobilunity, 検索日期：9月 20, 2025, https://mobilunity.com/blog/golang-vs-python/
30. The gopher can see you now. :: GoCV - Golang Computer Vision Using OpenCV 4, 検索日期：9月 20, 2025, https://gocv.io/
31. Build Raspberry Pi Zero W WiFi Camera using Go | go4vl - Medium, 検索日期：9月 20, 2025, https://medium.com/go4vl/build-a-wifi-camera-using-the-raspberry-pi-zero-w-a-camera-module-and-go-1d5fadfa7d76