**修正後完整規格書 (v1.1)**

**1️⃣ 專案總目標與核心理念**

**1.1 專案最終定位**

**打造企業級、金融業可用、長期穩定自動運行的 GPU 自動化量化交易訓練系統，專注於穩定性與長期複利增長，而非短期暴利。系統目標為：**

**穩定獲利，確保資金曲線平滑上升。**

**嚴格遵守風險控管，優先考慮穩定性高於報酬率。**

**以全自動化方式探索泛用最佳參數組合，適用於多市場與多時間框架。**

**提供可稽核、可復原、可長期運行的閉環系統，作為金融業量化交易的落地標準。**

**1.2 核心目標**

**🎯 長期穩定複利增長**

**目標為長期資金曲線穩定成長，非單次高報酬。**

**尋找各市場、各時間框架下的泛用型最佳參數，確保長期複利下資金穩定增長。**

**資金曲線需平滑，優先考慮穩定性（標準差<0.1）高於報酬率（Sharpe ≥ 1.5）。**

**🎯 多市場支援**

**支援以下市場：**

**幣安 USDT-M 合約：BTCUSDT、ETHUSDT、SOLUSDT、XRPUSDT、DOGEUSDT、ADAUSDT、BNBUSDT。**

**群益槓桿交易商 CFD 指數：NDAQ100、DJ30、JPN225。**

**未來市場新增需手動於設定檔中明確定義，未經批准不得自動加入市場。**

**🎯 多時間框架支援**

**支援時間框架：1分鐘（M1）、5分鐘（M5）、15分鐘（M15）、30分鐘（M30）、1小時（H1）、4小時（H4）、1天（D1）。**

**下單限制：僅限 BTCUSDT 的 15 分鐘框架（BTCUSDT\_15M）進行實際下單，使用幣安主網 API 取得 K 棒價格行為，幣安合約測試網 API 執行交易行為。**

**其他時間框架僅用於訓練與泛化參考，不進行實際下單。**

**🎯 模型多樣化**

**支援以下模型結構：**

**多層感知機（MLP）**

**卷積神經網路（CNN）**

**長短期記憶網路（LSTM）**

**變壓器（Transformer）**

**圖神經網路（GNN）**

**每種模型支援 Dropout 與 NoDropout 雙分流並行，確保泛化能力測試。**

**1.3 全流程閉環自動化**

**實現以下閉環流程：資料預處理 → 模型訓練（KFold） → 測試網模擬交易 → Optuna 超參數搜尋 → 強化學習（RL，DQN/PPO） → 模型與參數回寫 → 模擬測試 → 記錄績效 → 下一輪。**

**所有模組（訓練模組、測試網模組、推播模組、GUI、Optuna、RL）必須參與閉環，確保無縫串接。**

**測試網模組不僅用於驗證績效，還參與超參數搜尋與強化學習，形成閉環自適應調參。**

**採用無限輪迴方式持續訓練與測試，排除過度擬合風險，確保長期穩定性。**

**2️⃣ 交易目標、風控條件**

**2.1 資金與倉位管理**

**虛擬貨幣市場：**

**下單金額固定為總資金的 10%，透過幣安合約測試網 API 取得帳戶資金。**

**採用全倉模式，執行市價單，遵循 T+1 模式（除非遇強信號反手）。**

**CFD 市場：**

**使用群益經紀商規定的最小手數：**

**NDAQ100：0.1 手**

**DJ30：0.1 手**

**JPN225：1 手**

**採用全倉模式，執行市價單，遵循 T+1 模式（除非遇強信號反手）。**

**所有下單均需檢查可用保證金，確保不違反風控限制。**

**2.2 槓桿設定**

**虛擬貨幣市場：**

**BTCUSDT、ETHUSDT：125 倍槓桿**

**SOLUSDT：100 倍槓桿**

**XRPUSDT、DOGEUSDT、ADAUSDT、BNBUSDT：75 倍槓桿**

**CFD 市場：**

**NDAQ100、DJ30、JPN225：20 倍固定槓桿（可於設定檔調整）。**

**槓桿設定為固定值，不參與超參數搜尋或強化學習調整。**

**2.3 嚴格禁止行為**

**禁止加碼：已持有倉位時，不得額外增加同方向倉位。**

**禁止雙向持倉：不得同時持有多單與空單。**

**同一根 K 棒限制：**

**同向信號不得再次進場。**

**例外情況：當持有多單時遇到強信號空單（信號強度>0.9），可於同 K 棒立即平倉並反手開空；若同 K 棒再次出現強信號多單，可再次平空反手開多。**

**同 K 棒反手僅允許一次，避免反覆被洗單。**

**同 K 棒反手邏輯：**

**使用時間戳比較，確保同一 K 棒（BTCUSDT\_15M 為 15 分鐘內）僅允許一次反手。**

**若多次觸發反手，記錄異常並推播警示，格式如下：**

**【異常通知】**

**市場：BTCUSDT\_15M**

**事件：同 K 棒多次反手**

**動作：記錄異常並限制後續反手**

**時間：2025-07-02 16:45:00**

**反手觸發條件：信號強度>0.9，需經過訓練與 RL 動態調整。**

**2.4 停損停利與回撤規範**

**最大回撤： ≤ 25%（從資金高點至低點的跌幅）。**

**單筆交易最大虧損：初始預設 ≤ 2%（相對於初始資金），後續透過訓練、強化學習、強化自適應、超參數搜尋尋找最適合百分比，範圍 [0.01, 0.03]。**

**單日總損失：不限制，以信號為基準進行交易行為，不因單日虧損暫停交易。**

**爆倉保護：資金 ≤ 0 視為爆倉，立即停止當前輪次與市場的訓練/測試流程，並推播重大異常：**

**【重大異常：爆倉】**

**市場：BTCUSDT\_15M**

**資金餘額：0**

**原因：觸發爆倉保護**

**動作：立即停止測試網模擬交易**

**時間：2025-07-02 16:30:00**

**移動停損/停利：**

**移動停損百分比（Trailing Stop Loss）：由訓練與 RL 動態調整，預設範圍 [0.01, 0.05]。**

**移動停利百分比（Trailing Take Profit）：由訓練與 RL 動態調整，預設範圍 [0.02, 0.1]。**

**平本觸發（Breakeven Trigger）：由訓練與 RL 動態調整，預設範圍 [0.01, 0.05].**

**3️⃣ 資料預處理與快取**

**3.1 資料來源**

**3.1.1 資料更新頻率與維護流程**

**資料更新流程將完全自動化，由 n8n 工作流負責調度。n8n 將定時（例如每日凌晨 4 點）觸發一個 Python 數據管道腳本，該腳本會：**

**a. 透過 API 自動從幣安獲取增量 K 棒數據。**

**b. 觸發 MT5 內部的「總管 EA」，使其導出最新的增量 K 棒數據至其專用的 `MQL5\Files` 資料夾。**

**c. 數據管道腳本再對這兩個來源的原始數據進行統一的清洗與格式化，最終將乾淨、標準化的數據以「增量更新」的方式寫入中央 SQLite 資料庫。**

**3.1.2 資料庫與導出路徑定義**

**a. 所有處理後的乾淨數據，將統一儲存於中央 SQLite 資料庫中。資料庫檔案的路徑將在 `設定檔.py` 中定義。**

**b. MT5「總管 EA」導出數據的檔案路徑，將由 Python 的「數據管道模組」從 `設定檔.py` 中讀取並使用。**

**3.2 資料格式與欄位要求**

**必要欄位：**

**datetime：格式為 yyyy-mm-dd hh:mm:ss（UTC）。**

**open, high, low, close, volume：數值型，代表 K 棒開盤價、最高價、最低價、收盤價與成交量。**

**技術指標（若無則自動生成）：**

**SMA50：50 周期簡單移動平均，用於大趨勢濾網。**

**HMA\_16：16 周期赫爾移動平均，用於短期多空切換。**

**ATR\_14：14 周期平均真實波幅，用於波動度計算。**

**VHF\_28：28 周期垂直水平濾波器，用於趨勢強度。**

**Pivot High/Low：支撐與阻力點，周期由訓練決定（預設 5）。**

**技術指標生成：**

**若匯入資料缺少技術指標，由 資料預處理模組.py 自動生成。**

**生成邏輯參考 3.4.4 技術指標生成。**

**3.3 編碼自動偵測與轉換**

**支援編碼：UTF-8、UTF-8-SIG、Big5、GB2312。**

**匯入時自動偵測編碼（使用 chardet 套件，檢查前 10,000 位元組）。**

**若偵測到非標準編碼，自動轉換為 UTF-8 以供 GPU 加速計算。**

**轉換失敗記錄至 錯誤紀錄.db 並推播警示：**

**【異常通知】**

**市場：BTCUSDT\_15M**

**事件：編碼轉換失敗**

**檔案：BTCUSDT\_15M\_20250702.csv**

**動作：記錄異常並使用 UTF-8 重試**

**時間：2025-07-02 15:00:00**

**3.4 預處理流程**

**3.4.1 數據來源**

**主訓練流程在需要數據時，將直接向「資料庫管理模組.py」發出請求，獲取特定市場與時間框架的數據，而不是掃描本地檔案系統。所有掃描原始檔案、獲取、清洗、並寫入資料庫的髒活，已完全轉移給 3.1 節中定義的「數據管道模組」負責。**

**3.4.2 資料驗證**

**檢查欄位完整性：確保包含 open, high, low, close, volume, datetime。**

**檢查數據型別：確認 open, high, low, close, volume 為數值型，datetime 可解析為時間格式。**

**缺失值處理：**

**線性補值：對連續缺失值（少於 5 根 K 棒）進行線性插值。**

**平均補值：對非連續缺失值使用前後均值。**

**零補值：若無法補值，填充為 0 並記錄異常。**

**異常記錄至 錯誤紀錄.db 並推播警示。**

**3.4.3 分段讀取（防爆記憶體）**

**分批處理：每次處理 50,000 筆資料。**

**每批處理完畢後 sleep 5 毫秒，防止資源過載。**

**使用 GPU（CuPy/Torch）加速計算，若顯存不足自動切換至 CPU。**

**動態調整分批大小：**

**若 RAM 使用率 > 70%，分批大小減半（最低 10,000 筆）。**

**調整後記錄至日誌並推播通知：**

**【通知】**

**市場：BTCUSDT\_15M**

**事件：RAM 使用率過高，調整分批大小至 25000**

**時間：2025-07-02 15:05:00**

**3.4.4 技術指標生成**

**自動計算以下技術指標：**

**SMA50：50 周期簡單移動平均。**

**HMA\_16：16 周期赫爾移動平均，長度由訓練與 RL 動態調整（範圍 [8, 30]）。**

**ATR\_14：14 周期平均真實波幅，長度由訓練與 RL 動態調整（範圍 [7, 30]）。**

**VHF\_28：28 周期垂直水平濾波器，長度由訓練與 RL 動態調整（範圍 [14, 100]）。**

**Pivot High/Low：支撐與阻力點，周期由訓練與 RL 動態調整（範圍 [3, 20]）。**

**計算進度與用時記錄至日誌：日誌/預處理日誌\_YYYYMMDD.log。**

**指標計算失敗記錄至 錯誤紀錄.db 並推播警示。**

**3.4.5 分割訓練集/驗證集/測試集**

**此流程的數據來源，不再是分散的原始檔案，而是統一從 SQLite 資料庫中讀取。**

**預設比例：**

**訓練集：80%**

**驗證集：10%**

**測試集：10%**

**可於 設定檔.py 中調整比例。**

**分割結果儲存為：**

**快取/{市場}\_{週期}\_train.npz**

**快取/{市場}\_{週期}\_val.npz**

**快取/{市場}\_{週期}\_test.npz**

**3.4.6 壓縮並快取**

**處理後資料壓縮儲存為 NPZ 格式：**

**快取/{市場}\_{週期}\_已處理資料.npz**

**包含訓練集、驗證集、測試集的分割檔案。**

**壓縮使用 numpy.savez\_compressed 實現，確保高效儲存。**

**快取檔案生成後記錄至 預處理紀錄.db。**

**3.4.7 快取一致性檢查**

**原有的 `MD5 一致性檢查` 將被一個更高效的「時間戳比對」機制所取代。流程如下：**

**a. 當訓練模組需要數據時，它會先檢查對應的 `.npz` 快取檔案是否存在。**

**b. 如果存在，它會去查詢 SQLite 資料庫中，該商品的「最後更新時間戳」。**

**c. 如果資料庫中的數據，比現有的快取檔案還要新，則觸發快取重建：從資料庫讀取最新數據，重新進行「分割訓練集/驗證集/測試集」，並生成新的 `.npz` 快取檔案。**

**d. 如果資料庫中的數據沒有更新，則直接使用現有的快取檔案，跳過所有預處理步驟，極大地提升效率。**

**檢查失敗記錄至 錯誤紀錄.db 並推播警示：**

**【異常通知】**

**市場：BTCUSDT\_15M**

**事件：MD5 一致性檢查失敗**

**檔案：BTCUSDT\_15M\_20250702.csv**

**動作：重新處理檔案**

**時間：2025-07-02 15:10:00**

**3.5 GPU 加速策略**

**預設使用 GPU（CuPy/Torch）處理資料預處理與技術指標計算。**

**使用 torch.cuda.is\_available() 檢查 GPU 可用性。**

**顯存使用率 > 80% 時：**

**執行 torch.cuda.empty\_cache() 清理快取。**

**若清理後仍不足，自動切換至 CPU 執行。**

**切換記錄至日誌並推播通知：**

**【通知】**

**市場：BTCUSDT\_15M**

**事件：顯存不足，切換至 CPU 執行**

**時間：2025-07-02 15:15:00**

**3.6 SQLite 記錄（可稽核）**

**每次預處理結束，寫入資料庫：SQLite/預處理紀錄.db。**

**表名：預處理紀錄表。**

**欄位：**

**UUID：唯一識別碼（使用 uuid.uuid4() 生成）。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**來源檔案名：例如 BTCUSDT\_15M\_20250702.csv。**

**來源路徑：例如 D:/自動化交易/StrategyProject/訓練資料/。**

**筆數：處理的資料筆數。**

**欄位完整性：True/False，表示是否包含所有必要欄位。**

**處理時間：總處理時間（秒）。**

**GPU/CPU 標記：記錄使用 GPU 或 CPU 執行。**

**MD5：來源檔案與快取檔案的 MD5 碼。**

**快取檔名：例如 快取/BTCUSDT\_15M\_已處理資料.npz。**

**是否成功：True/False，表示處理是否成功。**

**重試次數：異常重試次數。**

**完成時間戳：格式為 yyyy-mm-dd hh:mm:ss。**

**3.7 推播同步**

**每次預處理完成後，自動推播通知：**

**【資料預處理完成】**

**UUID：xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx**

**市場：BTCUSDT**

**週期：15M**

**筆數：322000**

**GPU：是**

**快取：已完成**

**時間：2025-07-02 15:10:00**

**推播使用 Telegram Bot API，確保管理群收到即時更新。**

**若推播失敗，重試 5 次（每次間隔 5 秒），失敗後記錄至 日誌/推播錯誤\_YYYYMMDD.log。**

**3.8 錯誤處理與重試機制**

**觸發條件：**

**IO 阻塞（如檔案讀寫失敗）。**

**記憶體不足（RAM > 70% 或顯存 > 80%）。**

**資料格式錯誤（如欄位缺失、時間戳無效）。**

**處理邏輯：**

**自動重試 5 次，每次間隔 5 秒。**

**每次重試前執行 torch.cuda.empty\_cache()（若使用 GPU）。**

**重試失敗後：**

**記錄至 錯誤紀錄.db，包含錯誤訊息、Traceback 摘要、發生時間。**

**推播重大異常：**

**【重大異常】**

**市場：BTCUSDT\_15M**

**事件：資料預處理失敗，重試 5 次無效**

**動作：結束流程**

**時間：2025-07-02 15:20:00**

**結束當前市場/週期處理，進入下一市場/週期。**

**3.9 範例流程（以 BTCUSDT\_15M 為例）**

**將 BTCUSDT\_15M\_20250702.csv 放入 D:/自動化交易/StrategyProject/訓練資料/。**

**執行 訓練主控.py，自動呼叫 資料預處理模組.py。**

**GPU 加速處理，計算技術指標（SMA50、HMA\_16、ATR\_14、VHF\_28、Pivot High/Low）。**

**生成快取檔案：快取/BTCUSDT\_15M\_已處理資料.npz。**

**記錄至 SQLite/預處理紀錄.db，推播完成通知。**

**訓練模組自動讀取快取檔案，進入下一階段訓練。**

**4️⃣ 模型訓練流程、KFold、無限輪、超參數搜尋、強化學習、測試網閉環**

**4.1 訓練模組核心總原則**

**4.1.1 需參與訓練、超參數搜尋、強化學習的項目**

**以下項目必須進入訓練、Optuna 超參數搜尋、強化學習（DQN/PPO）與強化自適應（Adaptive RL），尋找最佳參數組合，不可硬編死值：**

**模型結構與訓練參數：**

**學習率（learning rate）：範圍 [1e-5, 1e-2]，對數尺度。**

**批次大小（batch size）：選項 [32, 64, 128, 256]。**

**Dropout 比率：範圍 [0.1, 0.5]（僅限使用 Dropout 的模型）。**

**隱藏層層數：範圍 [1, 4]。**

**每層神經元數：範圍 [64, 256]。**

**Optimizer 類型：選項 [Adam, SGD, RMSprop]。**

**技術指標參數：**

**SMA50：**

**是否啟用：True/False。**

**周期：範圍 [10, 200]，整數。**

**HMA\_16：**

**周期：範圍 [8, 30]，整數。**

**權重：範圍 [0.1, 1.0]，由 RL 動態調整。**

**ATR\_14：**

**周期：範圍 [7, 30]，整數。**

**用於移動停損/停利寬度，權重範圍 [0.1, 1.0]。**

**VHF\_28：**

**周期：範圍 [14, 100]，整數。**

**趨勢強度閾值：範圍 [0.1, 1.0]。**

**Pivot High/Low：**

**是否啟用：True/False。**

**周期：範圍 [3, 20]，整數。**

**下單與風控參數：**

**強信號閾值：範圍 [0.5, 0.95]，預設 0.9。**

**停損百分比：範圍 [0.01, 0.05]。**

**停利百分比：範圍 [0.02, 0.1]。**

**移動停損百分比：範圍 [0.01, 0.05]。**

**移動停利百分比：範圍 [0.02, 0.1]。**

**平本觸發：範圍 [0.01, 0.05]。**

**單筆交易最大虧損：範圍 [0.01, 0.03]，初始預設 2%，由訓練與 RL 動態調整。**

**信號觸發條件：**

**多指標交集權重：範圍 [0.1, 1.0]。**

**條件數量門檻：範圍 [1, 5]，整數。**

**模型推論信號閾值：範圍 [0.5, 0.9]。**

**測試網閉環參數：**

**模擬交易策略下單條件：由 RL 動態調整。**

**停損停利觸發條件：由 RL 動態調整。**

**強信號反手判定條件：由 RL 動態調整。**

**移動停損/停利啟用條件與參數：由 RL 動態調整。**

**技術指標參數搜尋範圍：**

**指標,參數,搜尋範圍,約束**

**SMA50,是否啟用,True/False,-**

**SMA50,周期,"[10, 200]",整數**

**HMA\_16,周期,"[8, 30]",整數**

**HMA\_16,權重,"[0.1, 1.0]",浮點數**

**ATR\_14,周期,"[7, 30]",整數**

**ATR\_14,權重,"[0.1, 1.0]",浮點數**

**VHF\_28,周期,"[14, 100]",整數**

**VHF\_28,趨勢強度閾值,"[0.1, 1.0]",浮點數**

**Pivot High/Low,是否啟用,True/False,-**

**Pivot High/Low,周期,"[3, 20]",整數**

**4.1.2 固定死的項目**

**以下項目為固定值，不參與訓練、超參數搜尋或強化學習：**

**最大回撤： ≤ 25%，若超過則停止流程並標記參數組合不可用。**

**單日總虧損：不限制，以信號為基準進行交易行為，不因單日虧損暫停交易。**

**爆倉條件：資金 ≤ 0，立即停止流程。**

**槓桿設定：**

**BTCUSDT、ETHUSDT：125 倍**

**SOLUSDT：100 倍**

**XRPUSDT、DOGEUSDT、ADAUSDT、BNBUSDT：75 倍**

**NDAQ100、DJ30、JPN225：20 倍**

**手續費率：**

**虛擬貨幣市場：0.05%（雙向）。**

**CFD 市場：0.01%（雙向）。**

**CFD 市場最小手數：**

**NDAQ100：0.1 手**

**DJ30：0.1 手**

**JPN225：1 手**

**4.2 訓練流程**

**4.2.1 模組主控**

**`訓練主控.py` 的角色，被明確定義為一個「可獨立運行的常駐服務 (Long-running Service)」。此服務由使用者手動啟動一次後，即進入「無限輪」訓練迴圈，其運行與 n8n 的數據更新流程完全解耦，不受其干擾。**

**4.2.2 資料加載**

**載入資料：**

**最新快取檔案：快取/{市場}\_{週期}\_train.npz、val.npz、test.npz。**

**最新最佳參數：SQLite/最佳參數記錄.db。**

**優先使用 GPU 訓練（檢查 torch.cuda.is\_available()），顯存不足自動切換至 CPU。**

**載入失敗記錄至 錯誤紀錄.db 並推播警示。**

**4.2.3 Optuna 超參數搜尋**

**搜尋所有「需參與搜尋的參數」（見 4.1.1）。**

**搜尋目標：**

**最大化 Sharpe（目標值 ≥ 1.5）。**

**最小化 Validation Loss。**

**控制最大回撤 ≤ 25%。**

**搜尋結果寫入：SQLite/最佳參數記錄.db。**

**每次搜尋完成推播通知：**

**【超參數搜尋完成】**

**市場：BTCUSDT\_15M**

**最佳參數：{learning\_rate: 0.001, dropout: 0.2, batch\_size: 64, ...}**

**Sharpe：1.72**

**最大回撤：19.3%**

**時間：2025-07-02 16:00:00**

**4.2.4 強化學習（RL）訓練**

**使用 DQN 或 PPO 作為強化學習核心。**

**Reward 計算：**

**正向 Reward：Sharpe（年化夏普比率）。**

**負向 Reward：最大回撤、資金曲線波動度（標準差）。**

**穩定性 Reward：資金曲線平滑度（標準差 < 0.1）。**

**RL 訓練結果寫入：SQLite/RL最佳策略.db。**

**表結構：**

**UUID：唯一識別碼。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**策略參數 JSON：儲存 RL 最佳策略參數。**

**Reward：總獎勵值。**

**Sharpe：年化夏普比率。**

**最大回撤：最大回撤百分比。**

**訓練輪次：當前訓練輪次。**

**記錄時間：格式為 yyyy-mm-dd hh:mm:ss。**

**4.2.5 模型訓練（KFold）**

**執行 5 折 KFold 交叉驗證。**

**使用最新超參數與 RL 策略進行訓練。**

**記錄以下指標：**

**Epoch 編號**

**訓練 Loss**

**驗證 Loss**

**F1 分數**

**Sharpe**

**最大回撤**

**資源使用率（GPU/CPU/RAM）**

**訓練結果寫入：SQLite/訓練記錄.db。**

**每次 Epoch 完成推播進度：**

**【訓練進度】**

**市場：BTCUSDT\_15M**

**Epoch：10/100**

**訓練 Loss：0.1234**

**驗證 Loss：0.0987**

**時間：2025-07-02 16:15:00**

**4.2.6 測試網閉環驗證**

**在此節中，明確區分兩種閉環驗證的路徑：**

**a. 幣安市場：維持不變，繼續使用「主網 API 獲取價格，合約測試網 API 執行交易」的直接連線方式。**

**b. MT5 (CFD) 市場：其閉環驗證是透過「雙向檔案通訊系統」完成。Python 引擎產生交易決策後，會寫入一個「指令檔案」；MT5 上的「總管 EA」讀取該指令並在模擬帳戶執行，再將執行結果寫入「結果檔案」，最後由 Python 讀取結果檔案以完成反饋閉環。**

**4.2.7 無限輪結構**

**定義：**

**一次完整流程（所有市場與時間框架的訓練+測試網驗證）為 1 次迭代。**

**10 次迭代為 1 輪。**

**終止條件：**

**Sharpe ≥ 1.5 且 最大回撤 ≤ 25%。**

**資金曲線標準差 < 0.1。**

**資源管理：**

**每輪結束後執行 torch.cuda.empty\_cache() 清理 GPU 記憶體。**

**檢查磁碟空間，若剩餘 < 10%，自動清理最舊快取檔案（保留 7 天）。**

**推播輪次完成通知：**

**【輪次完成】**

**輪次：第 3/10**

**市場：BTCUSDT\_15M**

**Sharpe：1.72**

**最大回撤：19.3%**

**時間：2025-07-02 16:45:00**

**若未滿足終止條件，進入下一輪訓練。**

**4.3 範例（實務執行流程）**

**預處理完成 BTCUSDT\_15M\_20250702.csv，生成快取檔案。**

**執行 訓練主控.py，讀取快取檔案。**

**執行 Optuna 超參數搜尋，找到最佳參數（如 learning\_rate=0.001, dropout=0.2, batch\_size=64）。**

**執行 RL 訓練，找到最佳策略（如強信號閾值 0.91、移動停損 2%、停利 5%）。**

**使用上述參數進行 5 折 KFold 訓練。**

**使用訓練完成的模型在測試網（BTCUSDT\_15M）執行模擬交易，K 棒價格行為從幣安主網 API 取得。**

**結果：**

**Sharpe：1.72**

**最大回撤：19.3%**

**滿足條件，標記參數組合為穩定可用，進入下一市場。**

**若未達標，記錄至 SQLite/訓練記錄.db，進入下一輪無限輪訓練。**

**5️⃣ 測試網模組與真實交易前安全驗證**

**5.1 測試網模組定位**

**在原有基礎上，新增以下描述：**

**a. 測試網模組將採用\*\*雙重執行路徑\*\*：對於幣安市場，它將直接透過 API 進行即時模擬交易；對於 MT5 (CFD) 市場，它將作為雙向檔案通訊系統的客戶端，透過讀寫「指令/結果檔案」來完成模擬交易閉環。**

**b. \*\*CFD 市場下單限制\*\*：測試網的 CFD 模擬交易，將\*\*僅限 NDAQ100 的 15 分鐘框架（NDAQ100\_15M）\*\*進行。所有其他 CFD 市場與時間框架，其數據僅用於模型的訓練與泛化能力參考，不執行任何模擬下單。**

**5.2 執行流程**

**5.2.1 模組**

**主控檔案：模擬交易模組.py、測試網模組.py。**

**5.2.2 環境初始化**

**使用幣安合約測試網 API（非主網）作為模擬下單介面，K 棒價格行為從幣安主網 API 取得。**

**載入：**

**最新訓練完成的模型權重（模型/{市場}\_{週期}\_model.pt）。**

**最新最佳參數（SQLite/最佳參數記錄.db）。**

**快取資料（快取/{市場}\_{週期}\_test.npz）。**

**預檢條件：**

**環境可連線（檢查幣安主網與測試網 API 連線狀態）。**

**快取資料可讀取。**

**GPU 可用（顯存使用率 < 80%）。**

**新增一條預檢條件：「檢查指令與結果檔案的讀寫路徑（從 `設定檔.py` 獲取）是否有效且具備存取權限。」**

**初始化失敗記錄至 錯誤紀錄.db 並推播警示。**

**5.2.3 資金與風控初始化**

**固定風控：**

**最大回撤 ≤ 25%**

**單筆虧損依 RL 調整，範圍 [0.01, 0.03]，初始預設 2%。**

**單日總虧損不限制，以信號為基準進行交易。**

**資金 ≤ 0 視為爆倉，立即停止流程。**

**動態參數：**

**強信號閾值、停損百分比、停利百分比、移動停損/停利、平本觸發等，由訓練與 RL 動態調整。**

**5.2.4 模擬交易邏輯**

**讀取當前 K 棒（從幣安主網 API 取得），使用訓練模型推論產生信號（Long / Short / No Action）。**

**信號符合以下條件才進行下單：**

**模型信號強度超過動態閾值（由 RL 調整，預設 0.9）。**

**滿足多指標交集（若啟用，例如 SMA50 與 HMA\_16 權重交集）。**

**不違反同 K 棒僅允許一次反手限制。**

**下單參數：**

**虛擬貨幣市場：固定總資金 10% 下單，資金透過幣安合約測試網 API 取得。**

**CFD 市場：使用最小手數（NDAQ100: 0.1 手、DJ30: 0.1 手、JPN225: 1 手）。**

**全倉市價單執行。**

**對於 CFD 市場的下單，將透過向指定的指令檔案路徑寫入交易指令（例如：`BUY,NDAQ100,0.1`）來完成。**

**持倉後執行：**

**停損（由 RL 調整，範圍 [0.01, 0.05]）。**

**停利（由 RL 調整，範圍 [0.02, 0.1]）。**

**移動停損/停利（由 RL 調整，範圍 [0.01, 0.05] / [0.02, 0.1]）。**

**爆倉判定（資金 ≤ 0），立即平倉、停止交易、推播異常。**

**\*\*MT5 延遲保護機制\*\*：為避免因檔案系統 I/O 延遲，導致系統在發出平倉指令後、尚未確認平倉完成前，就發出新的反向開倉指令，從而造成短暫的「假性雙向持倉」而違反風控規則。系統將在透過「指令檔案」發出一個平倉指令後，強制\*\*延遲等待一段時間\*\*（此延遲時間可在 `設定檔.py` 中設定，預設為 2 秒），然後才允許發出下一個新的開倉指令。**

**5.2.5 資金曲線與風控監控**

**每次交易記錄：**

**時間戳**

**信號方向（Long/Short/No Action）**

**進場價格**

**出場價格**

**損益**

**資金餘額**

**最大回撤**

**風控檢查：**

**若最大回撤 > 25%，立即停止流程。**

**若單筆虧損超出 RL 調整範圍 [0.01, 0.03]，觸發停損。**

**單日虧損不限制，僅以信號為基準交易。**

**記錄至 SQLite/測試網交易結果.db。**

**5.2.6 記錄與推播**

**交易結果記錄至：SQLite/測試網交易結果.db。**

**表結構：**

**UUID：唯一識別碼。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**模型：例如 MLP。**

**使用參數 UUID：對應最佳參數記錄。**

**Sharpe：年化夏普比率。**

**最大回撤：最大回撤百分比。**

**總收益率：總報酬率。**

**爆倉標記：True/False。**

**交易次數：總交易次數。**

**資金曲線檔案路徑：例如 圖片/BTCUSDT\_15M\_資金曲線\_20250702.html。**

**記錄時間：格式為 yyyy-mm-dd hh:mm:ss。**

**推播通知：**

**成功完成：推播績效摘要。**

**爆倉：立即推播重大異常。**

**超過風控限制：推播警示。**

**5.2.7 與訓練模組閉環互動**

**測試網結果作為下一輪訓練 Reward：**

**正向 Reward：Sharpe。**

**負向 Reward：最大回撤。**

**穩定性 Reward：資金曲線平滑度（標準差 < 0.1）。**

**若滿足穩定條件（Sharpe ≥ 1.5 且 最大回撤 ≤ 25%）：**

**標記參數組合為穩定可用。**

**保存至 最佳參數記錄.db。**

**進入下一市場或時間框架訓練。**

**若未達標，記錄結果並進入下一輪超參數搜尋與 RL 訓練。**

**5.3 爆倉與防呆安全保護**

**爆倉即停：**

**若資金 ≤ 0，立即平倉並停止流程。**

**記錄至 測試網交易結果.db，標記爆倉。**

**推播重大異常：**

**【重大異常：爆倉】**

**市場：BTCUSDT\_15M**

**資金餘額：0**

**原因：觸發爆倉保護**

**動作：立即停止測試網模擬交易**

**時間：2025-07-02 16:30:00**

**避免假停損回補：**

**爆倉後禁止重新加資繼續執行，模擬真實交易情境。**

**記錄爆倉事件至 錯誤紀錄.db。**

**防呆措施：**

**測試網模擬前檢查幣安主網與測試網 API 連線狀態、快取資料完整性、模型權重可用性。**

**異常觸發時自動重試 5 次（每次間隔 5 秒），失敗後停止流程並推播警示。**

**6️⃣ GUI 模組、推播模組、終端同步與防呆**

**6.1 模組定位**

**GUI 模組：提供訓練、測試網、資料預處理的即時狀態監控與人工控制介面。**

**推播模組：透過 Telegram Bot API 提供進度、完成、異常的即時通知。**

**終端同步顯示：作為 GUI 與推播的後備，確保狀態可見性。**

**目標：**

**實現所有模組（訓練、測試網、資料預處理）的狀態、進度、異常即時同步顯示。**

**不干擾自動化流程，即使 GUI 或推播失效，主流程繼續運行。**

**GUI 提供人工緊急控制（暫停、恢復、釋放記憶體、查看紀錄）。**

**完善防呆機制，確保系統穩定運行。**

**6.2 GUI 模組詳細設計**

**6.2.1 技術棧**

**使用 PySimpleGUI 或 Tkinter 實現。**

**支援未來切換至 Electron 或 WebView，需預留轉換計畫：**

**轉換步驟：將 PySimpleGUI/Tkinter 介面重寫為 HTML/CSS/JavaScript，整合 Electron 或 WebView。**

**預計成本：約 2-3 週開發時間，需 1 名前端工程師。**

**自動調整解析度與 DPI，適應不同螢幕。**

**即時顯示 GPU/CPU/RAM 使用率與磁碟剩餘空間。**

**6.2.2 功能模組**

**進度與狀態監控：**

**模型訓練：**

**顯示當前訓練市場（例如 BTCUSDT）、週期（例如 15M）、模型（例如 MLP）。**

**顯示輪次（第 N/10 次）與當前 Epoch（例如 10/100）。**

**測試網模擬：**

**顯示模擬交易中的交易筆數。**

**動態更新資金曲線、Sharpe、最大回撤。**

**資料預處理：**

**顯示當前處理檔案、市場、週期、進度（例如 50%）。**

**資源監控：**

**GPU 使用率與顯存使用率。**

**CPU 使用率。**

**RAM 使用率。**

**磁碟剩餘空間（D:/自動化交易/StrategyProject）。**

**視覺化：**

**即時更新資金曲線圖（使用 Plotly 生成交互式圖表）。**

**顯示 Sharpe 與最大回撤的動態趨勢圖。**

**訓練與測試進度條（百分比顯示）。**

**控制面板：**

**暫停按鈕：暫停訓練或測試流程，記錄中斷點至 錯誤紀錄.db。**

**繼續按鈕：從中斷點恢復流程。**

**釋放記憶體按鈕：執行 torch.cuda.empty\_cache() 與垃圾回收。**

**即時查看 SQLite 最新紀錄按鈕：**

**查詢範圍：最近 100 筆預處理、訓練、交易記錄。**

**顯示格式：表格（包含 UUID、市場、週期、時間戳等）或交互式圖表（資金曲線、Sharpe 趨勢）。**

**異常處理：查詢失敗時顯示錯誤訊息並記錄至 錯誤紀錄.db。**

**切換 CPU/GPU 執行：僅限測試模式，手動切換執行設備。**

**GUI 查詢功能：**

**查詢最近 100 筆 預處理紀錄.db、訓練記錄.db、測試網交易結果.db 記錄。**

**顯示格式：**

**表格：包含 UUID、市場、週期、關鍵指標（Sharpe、回撤、Loss 等）、時間戳。**

**圖表：資金曲線、Sharpe 趨勢、回撤趨勢（使用 Plotly 實現）。**

**異常處理：**

**若查詢失敗，顯示錯誤訊息（如「資料庫連線失敗」）。**

**記錄至 錯誤紀錄.db 並推播警示：**

**【異常通知】**

**市場：BTCUSDT\_15M**

**事件：GUI 查詢 SQLite 失敗**

**動作：記錄異常並重試**

**時間：2025-07-02 16:50:00**

**6.2.3 更新頻率**

**資源監控：每 3 秒更新。**

**圖表更新：每 30 秒更新（可於 設定檔.py 調整）。**

**防呆限制：**

**更新失敗不影響主流程，記錄至 錯誤紀錄.db。**

**GUI 異常自動重啟 5 次（每次間隔 5 秒），失敗後關閉 GUI，主流程繼續運行。**

**6.3 推播模組詳細設計**

**6.3.1 技術棧**

**使用 Telegram Bot API（透過 python-telegram-bot 實現）。**

**敏感憑證（Bot Token、Chat ID）儲存於 .env.txt，使用 os.getenv 載入。**

**6.3.2 推播類型**

**進度推播：**

**每完成 10% 進度或每輪結束時推播：**

**【進度通知】**

**市場：BTCUSDT\_15M**

**輪次：第 3/10**

**Sharpe：1.31**

**最大回撤：17.5%**

**時間：2025-07-02 16:45:00**

**完成推播：**

**訓練或測試完成後推播：**

**【訓練完成】**

**市場：BTCUSDT\_15M**

**Sharpe：1.72**

**最大回撤：19.3%**

**達標：是**

**已儲存參數至：最佳參數記錄.db**

**時間：2025-07-02 17:00:00**

**異常推播：**

**爆倉、超過風控限制、資源不足、異常中止：**

**【重大異常】**

**市場：BTCUSDT\_15M**

**事件：資金歸零（爆倉）**

**動作：立即停止流程**

**時間：2025-07-02 17:05:00**

**6.3.3 推播頻率控制**

**限制：**

**每分鐘推播次數不超過 10 次，符合 Telegram API 限制。**

**使用 PushLimiter 類實現頻率控制，記錄最近 60 秒內的推播次數。**

**異常處理：**

**若超過頻率限制，緩存訊息至 SQLite/push\_cache.db，每 5 分鐘重試。**

**重試失敗記錄至 日誌/推播錯誤\_YYYYMMDD.log。**

**6.3.4 異常容錯**

**若推播失敗（斷網或 Telegram API 限制）：**

**自動重試 5 次（每次間隔 5 秒）。**

**重試失敗後：**

**記錄至 日誌/推播錯誤\_YYYYMMDD.log。**

**緩存至 SQLite/push\_cache.db，每 5 分鐘重試。**

**主流程繼續執行，不受推播失敗影響。**

**6.4 終端同步顯示**

**所有推播內容同步列印至終端（stdout），作為 GUI 與推播的後備。**

**列印格式：**

**[推播][進度] BTCUSDT\_15M 第 3/10 輪 Sharpe: 1.31 Max Drawdown: 17.5% [2025-07-02 16:45:00]**

**重要異常以醒目格式顯示：**

**!!! 異常 !!!**

**市場：BTCUSDT\_15M**

**事件：資金歸零（爆倉）**

**動作：立即停止流程**

**時間：2025-07-02 17:05:00**

**6.5 防呆機制**

**推播模組、GUI 模組、終端輸出均符合：**

**推播或 GUI 失效不影響主流程，主流程繼續運行。**

**GUI 異常自動重啟 5 次（每次間隔 5 秒），失敗後關閉 GUI，釋放資源。**

**若因記憶體不足導致 GUI 當機，自動關閉 GUI，執行 torch.cuda.empty\_cache()。**

**終端列印作為最終後備，確保狀態永遠可見。**

**7️⃣ SQLite 資料庫架構、備份與復原策略**

**7.1 SQLite 資料庫用途**

**提供快速查詢歷史資料、訓練結果、測試網績效、超參數搜尋結果、強化學習結果。**

**確保資安異動與錯誤紀錄的可稽核性。**

**支援 GUI 即時查看最新資料庫狀態。**

**作為異常恢復後無縫銜接流程的依據，避免重頭運行。**

**7.2 SQLite 資料庫結構**

**資料庫存放路徑：SQLite/\*.db。**

**以下為必要表結構，嚴禁添加未經許可的表以避免混亂。**

**7.2.1 預處理紀錄資料庫**

**檔名：SQLite/預處理紀錄.db**

**表名：預處理紀錄表**

**欄位：**

**UUID：唯一識別碼（使用 uuid.uuid4() 生成）。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**來源檔案名：例如 BTCUSDT\_15M\_20250702.csv。**

**來源路徑：例如 D:/自動化交易/StrategyProject/訓練資料/。**

**筆數：處理的資料筆數。**

**欄位完整性：True/False，表示是否包含所有必要欄位。**

**處理時間：總處理時間（秒）。**

**GPU/CPU 標記：記錄使用 GPU 或 CPU 執行。**

**MD5：來源檔案與快取檔案的 MD5 碼。**

**快取檔名：例如 快取/BTCUSDT\_15M\_已處理資料.npz。**

**是否成功：True/False，表示處理是否成功。**

**重試次數：異常重試次數。**

**完成時間戳：格式為 yyyy-mm-dd hh:mm:ss。**

**7.2.2 最佳參數記錄資料庫**

**檔名：SQLite/最佳參數記錄.db**

**表名：最佳參數表**

**欄位：**

**UUID：唯一識別碼。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**模型：例如 MLP。**

**Dropout 設定：True/False。**

**學習率：例如 0.001。**

**批次大小：例如 64。**

**層數：例如 2。**

**Optimizer：例如 Adam。**

**訓練 Loss：訓練集損失。**

**驗證 Loss：驗證集損失。**

**Sharpe：年化夏普比率。**

**最大回撤：最大回撤百分比。**

**記錄時間：格式為 yyyy-mm-dd hh:mm:ss。**

**7.2.3 測試網交易結果資料庫**

**檔名：SQLite/測試網交易結果.db**

**表名：測試結果表**

**欄位：**

**UUID：唯一識別碼。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**模型：例如 MLP。**

**使用參數 UUID：對應最佳參數記錄。**

**Sharpe：年化夏普比率。**

**最大回撤：最大回撤百分比。**

**總收益率：總報酬率。**

**爆倉標記：True/False。**

**交易次數：總交易次數。**

**資金曲線檔案路徑：例如 圖片/BTCUSDT\_15M\_資金曲線\_20250702.html。**

**記錄時間：格式為 yyyy-mm-dd hh:mm:ss。**

**7.2.4 RL 策略結果資料庫**

**檔名：SQLite/RL最佳策略.db**

**表名：策略表**

**欄位：**

**UUID：唯一識別碼。**

**市場：例如 BTCUSDT。**

**週期：例如 15M。**

**策略參數 JSON：儲存 RL 最佳策略參數（JSON 格式）。**

**Reward：總獎勵值。**

**Sharpe：年化夏普比率。**

**最大回撤：最大回撤百分比。**

**訓練輪次：當前訓練輪次。**

**記錄時間：格式為 yyyy-mm-dd hh:mm:ss。**

**7.2.5 錯誤紀錄資料庫**

**檔名：SQLite/錯誤紀錄.db**

**表名：錯誤紀錄表**

**欄位：**

**UUID：唯一識別碼。**

**模組：例如 訓練主控、測試網模組。**

**事件：例如 資金歸零、記憶體不足。**

**錯誤訊息：截斷摘要（最多 1000 字元）。**

**Traceback 摘要：異常堆疊摘要（最多 1000 字元）。**

**發生時間：格式為 yyyy-mm-dd hh:mm:ss。**

**是否已處理：True/False。**

**7.3 資料庫異動管理與稽核**

**所有異動將記錄至一個專門的 SQLite 資料庫表格中（例如在 `預處理紀錄.db` 中建立一個名為 `異動歷程表` 的 Table），以提升穩定性與查詢效率。**

**稽核要求：**

**所有異動可追溯，支援 GUI 查詢。**

**異動記錄保留 30 天，自動清理舊記錄。**

**7.4 備份策略**

**每日備份流程，將由一個獨立的 n8n 工作流來實現。n8n 在每日凌晨 03:00 被觸發，執行一個 Python 腳本將所有必要檔案壓縮打包，並可選擇性地將備份檔上傳至獨立的雲端儲存，實現異地備份。**

**備份項目：**

**SQLite/\*.db（所有資料庫檔案）。**

**模型/（模型權重檔案）。**

**JSON/（參數快取檔案）。**

**.env.txt（敏感設定檔案）。**

**設定檔.py（系統設定檔案）。**

**dot/（依賴圖與異動紀錄）。**

**備份路徑：**

**備份/YYYYMMDD\_HHMM/。**

**滾動保留：**

**保留最近 30 天備份，自動刪除最舊備份。**

**若磁碟剩餘空間 < 10%，立即清理最舊備份並推播通知：**

**【通知】**

**事件：磁碟剩餘空間不足，清理最舊備份**

**備份路徑：備份/20250601\_0300/**

**時間：2025-07-02 03:01:00**

**推播通知：**

**備份完成：**

**【備份完成】**

**位置：備份/20250702\_0300/**

**包含：SQLite、模型、設定**

**時間：2025-07-02 03:01:00**

**備份失敗：**

**重試 3 次（每次間隔 5 秒）。**

**失敗後推播重大異常：**

**【重大異常】**

**事件：備份失敗，重試 3 次無效**

**動作：記錄異常並中止備份**

**時間：2025-07-02 03:05:00**

**7.5 復原策略**

**7.5.1 復原流程步驟**

**檢查最近一次有效備份（備份/YYYYMMDD\_HHMM/）。**

**複製備份檔案至原始路徑：**

**SQLite/\*.db → SQLite/。**

**模型/ → 模型/。**

**JSON/ → JSON/。**

**.env.txt → D:/自動化交易/StrategyProject/。**

**設定檔.py → D:/自動化交易/StrategyProject/。**

**dot/ → dot/。**

**驗證檔案完整性（使用 MD5 碼比對）。**

**載入最新 SQLite/\*.db，恢復歷史紀錄與進度。**

**載入最新模型權重（模型/{市場}\_{週期}\_model.pt），支持續訓。**

**恢復 GUI 與推播模組狀態，確保監控正常。**

**執行 訓練主控.py，從中斷點無縫恢復。**

**異常處理：**

**若備份檔案損壞，使用次新備份（備份/YYYYMMDD\_HHMM-1/）。**

**若所有備份均不可用，推播重大異常：**

**【重大異常】**

**事件：所有備份檔案損壞**

**動作：停止恢復流程，請檢查備份**

**時間：2025-07-02 03:10:00**

**記錄至 錯誤紀錄.db。**

**7.5.2 資安補充**

**備份包含敏感資料（.env.txt），需儲存於內網或加密硬碟，嚴禁上傳公開空間。**

**備份與復原過程記錄至：日誌/備份紀錄\_YYYYMMDD.log。**

**若備份或復原失敗，推播通知管理群：**

**【重大異常】**

**事件：備份/復原失敗**

**動作：記錄異常並中止流程**

**時間：2025-07-02 03:15:00**

**8️⃣ 最終整體防呆、自癒、資安總結封包**

**8.1 通用規則補充**

**a. 金融業標準：所有未在規格書中明確定義的細節，都將以「金融業用量化交易訓練程式」的專業標準為標竿進行設計與實現。**

**b. 防呆機制：所有模組的設計，都必須包含完善的防呆機制（如 `try-except` 異常捕捉、函式輸入參數的型別與範圍檢查、資源佔用監控等），以防止任何可預期的錯誤導致系統崩潰，確保長期運行的穩定性。**

**8.2 自癒與降載機制**

**8.2.1 自癒標準流程**

**一個獨立的 n8n 工作流將負責「系統健康度巡檢」。該工作流定時觸發一個輕量 Python 腳本回報系統資源狀態。若在主訓練流程開始前資源已過高，n8n 可直接決定不啟動任務並發送警報。**

**異常偵測：**

**檢測 CPU > 85%、GPU 顯存 > 80%、RAM > 70%、磁碟剩餘 < 10%。**

**檢測 IO 阻塞、網路斷線、資料庫錯誤等異常。**

**推播通知：**

**推播自癒開始：**

**【通知】**

**市場：BTCUSDT\_15M**

**事件：檢測到記憶體不足，正在進行自癒**

**時間：2025-07-02 17:00:00**

**執行降載與資源釋放：**

**降低批次大小：50,000 → 20,000 → 10,000 → 5,000。**

**降低多進程併發數：4 → 2 → 1。**

**執行 torch.cuda.empty\_cache() 清理 GPU 記憶體。**

**暫停非必要模組（GUI 即時繪圖、推播頻率降低）。**

**重試流程：**

**自動重試 5 次（每次間隔 5 秒）。**

**每次重試前檢查資源狀態，確保可用。**

**結果處理：**

**成功恢復：流程繼續，推播通知：**

**【通知】**

**市場：BTCUSDT\_15M**

**事件：自癒成功，流程恢復**

**時間：2025-07-02 17:05:00**

**失敗：中止該模組流程，推播重大異常：**

**【重大異常】**

**市場：BTCUSDT\_15M**

**事件：自癒失敗，重試 5 次無效**

**動作：中止流程**

**時間：2025-07-02 17:10:00**

**8.3 資安總結**

**8.3.1 敏感資料加密流程**

**將建立一個專門的「資安管理模組.py」，獨佔所有加解密操作。其他模組僅透過標準的 `os.getenv()` 來獲取憑證，實現安全機制的完全封裝和隔離。**

**敏感資料儲存：**

**所有敏感資料（API Key、Telegram Bot Token、Chat ID）儲存於 .env.txt。**

**使用 cryptography.fernet.Fernet 加密 .env.txt：**

**生成金鑰並儲存於內網安全位置（D:/自動化交易/StrategyProject/secure\_key.key）。**

**加密後檔案儲存於：D:/自動化交易/StrategyProject/.env.encrypted。**

**解密流程：**

**from cryptography.fernet import Fernet**

**with open("secure\_key.key", "rb") as key\_file: key = key\_file.read()**

**cipher = Fernet(key)**

**with open(".env.encrypted", "rb") as enc\_file: decrypted = cipher.decrypt(enc\_file.read())**

**嚴禁將 .env.txt 或金鑰上傳至公開雲端或 Git。**

**憑證載入：**

**使用 os.getenv 載入敏感憑證：**

**import os**

**api\_key = os.getenv("BINANCE\_API\_KEY")**

**憑證更新流程：**

**每月更新 API 金鑰與 Telegram Bot Token。**

**更新記錄至：日誌/資安事件\_YYYYMMDD.log。**

**格式：**

**[資安事件] 更新 BINANCE\_API\_KEY**

**時間：2025-07-02 10:00:00**

**原值：xxxx（前 4 字元）**

**新值：yyyy（前 4 字元）**

**原因：定期更新**

**日誌與推播保護：**

**日誌、推播與 GUI 不顯示完整憑證或 Token，僅顯示前 4 字元。**

**異常退出處理：**

**流程異常退出時：**

**清理快取檔案（快取/\*.npz）。**

**清理 GPU 記憶體（torch.cuda.empty\_cache()）。**

**關閉測試網 API 連線。**

**推播通知：**

**【重大異常】**

**事件：流程異常退出**

**動作：清理快取、釋放 GPU 記憶體、關閉 API 連線**

**時間：2025-07-02 17:15:00**

**8.4 爆倉與極端風控防護**

**系統將提供一個由 n8n Webhook 驅動的「一鍵中止 (Kill-Switch)」機制。觸發時，n8n 會創建 `KILL.flag` 檔案，所有常駐服務（`訓練主控.py` 和 MT5 的『總管 EA』）均需檢查此旗標並執行安全停機程序。**

**爆倉處理：**

**資金 ≤ 0 時：**

**強制平倉（模擬與測試網）。**

**記錄至 SQLite/測試網交易結果.db，標記爆倉。**

**推播重大異常：**

**【重大異常：爆倉】**

**市場：BTCUSDT\_15M**

**資金餘額：0**

**原因：觸發爆倉保護**

**動作：立即停止流程**

**時間：2025-07-02 17:05:00**

**停止當前市場、時間框架、參數組合的訓練與測試流程。**

**禁止回補資金：**

**爆倉後不可重新加資繼續測試，模擬真實交易情境。**

**記錄至 錯誤紀錄.db，包含爆倉原因與參數組合。**

**8.5 長期穩定運行要求**

**8.5.1 長期穩定性監控**

**監控頻率：**

**每 10 輪訓練檢查一次穩定性指標：**

**Sharpe ≥ 1.5**

**最大回撤 ≤ 25%**

**資金曲線標準差 < 0.1**

**指標追蹤方式：**

**從 SQLite/測試網交易結果.db 提取最近 100 筆記錄。**

**生成長期趨勢圖（資金曲線、Sharpe、回撤），儲存至：**

**圖片/{市場}\_{週期}\_長期趨勢\_YYYYMMDDHHMMSS.html**

**推播趨勢圖通知：**

**【通知】**

**市場：BTCUSDT\_15M**

**事件：生成長期趨勢圖**

**路徑：圖片/BTCUSDT\_15M\_長期趨勢\_20250702170000.html**

**時間：2025-07-02 17:00:00**

**異常處理：**

**若穩定性指標異常（標準差 > 0.1 或 Sharpe < 1.5），暫停流程並推播警示：**

**【異常通知】**

**市場：BTCUSDT\_15M**

**事件：長期穩定性異常，標準差 0.15**

**動作：暫停流程，等待檢查**

**時間：2025-07-02 17:10:00**

**記錄至 錯誤紀錄.db。**

**其他要求：**

**GPU 可用率高於 50%，顯存使用率 < 80%.**

**每日備份（凌晨 03:00），保留 30 天滾動備份。**

**異動記錄至：dot/異動歷程\_YYYYMMDD.xlsx。**

**爆倉不可回補資金，必須停機檢查。**

**8.6 可實務落地的最終母規**

**本規格書已：**

**完整吸收 GPT 與 Grok 版本的優點。**

**整合專案歷來所有規則。**

**嚴格區分「固定死」與「需訓練、超參數搜尋、RL 自適應」的項目。**

**達到金融業量化交易落地專案標準。**

**可直接存入：**

**Notion**

**Obsidian**

**私有內網知識庫**

**作為永久、穩定、唯一參考母檔，確保未來任何 GPT、Grok 或開發者接手均無上下文遺漏或誤解。**