# AIVA 最新版本 AI 閉環功能檢查報告

## 1. BioNeuron 攻擊計畫產生功能

根據最新的程式碼，BioNeuron（BioNeuronRAGAgent）的**攻擊計畫生成**目前尚未完全實作為動態生成。程式架構雖然預留了藉由 RAG 引擎和神經網路決策核心產生計畫的機制，但**現階段仍採用預定義樣板或場景中的固定計畫**。例如，在 TrainingOrchestrator 中有註解明確指出：「*這裡應該調用 AI 模型生成計畫，現在使用場景中的預定義計畫*」[[1]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L107-L115)。也就是說，目前系統並未根據使用者輸入即時創建全新的 AST 攻擊流程圖，而是**使用預先定義的攻擊計畫模板**（如情境 Scenario 內建的計畫）來進行後續步驟[[1]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L107-L115)[[2]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L117-L125)。因此在使用者輸入目標或任務後，BioNeuron **尚不能真正自主地產生完全動態的攻擊計畫**；目前的實作偏向使用固定範本或簡單的占位符計畫（例如 Master Controller 裡決策結果的 "plan": None 並標註 TODO）[[3]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/core/aiva_core/bio_neuron_master.py#L331-L339)。

**結論：** BioNeuron 產生攻擊計畫的功能在架構上已佈局，但最新程式碼顯示**仍未實現真正的動態計畫生成**。開發者需進一步完成該部分（取代暫時的樣板計畫），才能讓 AI 根據使用者輸入產生具體的攻擊步驟方案。

## 2. 攻擊計畫執行方式與實際執行程度

AIVA 專案內已實作 **AttackOrchestrator**（攻擊編排器）與 **PlanExecutor**（計畫執行器）來協調攻擊計畫的執行。然而，**目前的執行仍以模擬為主，未真正對外部目標進行攻擊操作**。從程式碼細節可看出：

* **AttackOrchestrator** 負責將 AST 攻擊流程轉換為執行計畫（任務序列和工具決策）。**PlanExecutor** 則按照執行計畫逐步執行任務並記錄過程[[4]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L99-L107)[[5]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L119-L127)。架構上確實透過這兩個元件來運行攻擊計畫。
* **實際執行模式：** PlanExecutor 在執行任務時，會將任務訊息透過 RabbitMQ 傳遞給對應功能模組，但**接收結果的機制尚未完成**。程式中的 \_wait\_for\_result 明確標示 **TODO**，當前只是等待一段時間後**回傳模擬結果**[[6]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L382-L390)：「暫時返回模擬結果」。這表示即使任務經由 MQ 發出，最終還是用預設的假資料做為結果。
* **模擬執行：** 另一方面，專案也實作了 TaskExecutor 來直接執行任務，並與 ExecutionMonitor 整合進行追蹤。在 TaskExecutor 中，各類任務的執行邏輯目前**全部以 Mock 實現**，並未真正呼叫外部掃描或攻擊工具[[7]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L128-L136)[[8]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L160-L169)。例如 \_execute\_scan\_service、\_execute\_function\_service 等方法都產生**假想的結果**（如隨機偵測到漏洞的回傳)[[8]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L160-L169)[[9]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L200-L208)。程式碼註明「*實際與各服務整合 TODO，目前使用 Mock 實現*」[[10]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L128-L135)。

綜上所述，**AttackOrchestrator + PlanExecutor 的串接是有的，但實際執行仍處於模擬階段**。攻擊計畫會經由 Orchestrator 轉成任務並交由 PlanExecutor 執行，但 PlanExecutor 並未真正等待來自真實工具的響應，而是立即給出模擬完成的結果[[6]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L382-L390)。同樣地，若走 TaskExecutor 路線，則每一步都用假資料回傳[[8]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L160-L169)[[9]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L200-L208)。**目前並沒有對真實目標系統執行攻擊，只是模擬流程**。未來需完成 RabbitMQ 結果接收及工具服務整合，才能將模擬替換為真實執行。

## 3. 執行過程的 Trace、評估結果與經驗樣本產生

最新版本程式碼已將**執行追蹤（Execution Trace）**、**計畫與執行對比評估**以及**經驗樣本存儲**等閉環學習要素串接起來，大部分功能已有基本實作：

* **執行 Trace 紀錄：** ExecutionMonitor 和 TraceRecorder 模組會在每次任務執行時記錄關鍵事件，包括任務開始、每個步驟（例如工具呼叫）和任務結束/錯誤等。執行計畫開始時建立一個 trace session，隨後對每個任務記錄輸入參數、輸出結果以及執行狀態[[11]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L114-L123)[[12]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L150-L158)。在 PlanExecutor 或 TaskExecutor 執行任務過程中，也透過 ExecutionMonitor 不斷寫入這些追蹤資訊[[13]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L233-L241)[[14]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L166-L174)。最終在計畫完成時，ExecutionMonitor 會**完成並彙總整個 ExecutionTrace**，其中包含所有步驟的執行記錄（entries）[[15]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L248-L257)[[16]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L259-L267)。
* **結果評估 (AST對比)：** 專案實作了 ASTTraceComparator 來對比預期的 AST 攻擊流程與實際執行的 Trace。執行完畢後，BioNeuron 會調用 Comparator 的 compare 方法計算**多項評估指標**，如**完成率**（完成步驟數/預期步驟數）、**順序匹配率**、**成功/失敗步驟數**、**錯誤數**等，並綜合計算一個 overall score[[17]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/analysis/ast_trace_comparator.py#L98-L106)[[18]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/analysis/ast_trace_comparator.py#L124-L132)。同時還可產生**回饋訊息（feedback）**，作為強化學習的獎勵信號。上述比較與回饋在 BioNeuron 執行攻擊計畫的流程中已被使用[[19]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L476-L485)。
* **經驗樣本生成與存儲：** 一旦取得執行 Trace 和比較指標，系統會將此次計畫執行視為一個「經驗」。在 BioNeuron 核心實作中，如果啟用了 Experience Repository，代碼會將**AST 攻擊圖、執行 Trace、評估 metrics 以及 feedback**等打包，透過 ExperienceRepository.save\_experience() 保存到經驗資料庫[[20]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L480-L489)。保存內容包括計畫ID、攻擊類型、目標資訊，以及各項評估結果和回饋[[21]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L481-L489)。例如上述程式碼將 overall\_score 等指標與 trace 全部序列化後存入資料庫記錄[[22]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L82-L91)[[23]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L98-L106)。另外，在訓練模式下，TrainingOrchestrator 也有類似邏輯收集每次執行的 ExperienceSample 並添加到經驗管理器，只是該部分詳盡的樣本提取（如獎勵計算）仍標註 TODO[[24]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L183-L189)。
* **經驗用於訓練：** 所有累積的經驗記錄可進一步提供給模型訓練機制。ExperienceRepository 支援按攻擊類型或分數篩選經驗資料[[25]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L138-L147)[[26]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L160-L169)。在 BioNeuron 核心中，有 train\_from\_experiences() 方法會從經驗庫抓取一定量的高分經驗樣本進行訓練[[27]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L530-L538)[[28]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L539-L546)。也就是說，**執行產生的經驗確實可以迴饋到學習模組**。目前此流程需要手動觸發，但已打通「經驗累積 → 模型學習」的資料管道。

**結論：** 執行過程的追蹤、評估與經驗儲存功能**基本完備**。每次攻擊計畫執行都會產生**完整的 ExecutionTrace**，並透過 AST 對比得到評估結果（Completion rate、Overall score 等）。這些結果與 Trace 一同構成**經驗樣本**，保存於經驗庫中，可供後續模型訓練使用[[20]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L480-L489)。目前經驗提取和回饋計算部分雖有簡化，但框架已具備閉環所需的資料收集與評價機制。

## 4. 模型訓練更新對 BioNeuron 輸出的影響

最新版本中，**模型訓練機制已經實作，並且能夠更新 BioNeuron 的決策模型參數**，從而改變其日後的推理輸出。具體來說：

* BioNeuron 的決策核心模型為 ScalableBioNet（約500萬參數的神經網路）。在程式中，訓練由 ModelUpdater 協調，結合 ExperienceDataLoader 和 ModelTrainer 來對 ScalableBioNet 進行微調訓練[[29]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L80-L88)。ExperienceDataLoader 會從經驗庫中載入樣本特徵 X 和標籤 y（以經驗的 overall\_score 等為學習目標）[[30]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/data_loader.py#L48-L56)[[31]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/data_loader.py#L80-L88)。
* **參數更新：** ModelUpdater.update\_from\_recent\_experiences() 會抓取最近的經驗資料，切分訓練集/驗證集後，調用 ModelTrainer.train() 進行梯度下降訓練[[29]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L80-L88)。程式碼中雖採用簡化的訓練算法（均方誤差損失、部分權重更新等），但確實在迴圈中**修改了模型的權重值**（例如更新 fc1 層的權重）[[32]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/trainer.py#L218-L225)。訓練完成後，系統會記錄最終的 loss 和 accuracy，並打印「*Model update completed*」日誌確認模型更新成功[[33]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L109-L117)。模型權重也會透過 \_save\_model() 儲存下來，以備後續載入或版本管理[[34]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L84-L93)[[35]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L189-L198)。
* **輸出改變：** 由於 BioNeuron 的決策是依賴於其神經網路（ScalableBioNet）的 forward 輸出，在參數更新後，同樣輸入條件下網路輸出機率分佈將發生變化。換言之，**BioNeuron 之後產生的決策（例如工具選擇或計畫傾向）會隨著訓練經驗的累積而調整**。雖然目前訓練觸發需要手動呼叫，但機制上只要經驗樣本足夠並進行訓練，多次迭代後BioNeuron的推理結果理論上會逐步優化。

需要注意的是，現階段的模型訓練主要偏監督學習（基於經驗分數），強化學習管道也有雛形（如 train\_reinforcement 方法），但完整的獎勵計算與策略更新尚未完善[[36]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/learning/model_trainer.py#L140-L149)[[37]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/learning/model_trainer.py#L156-L165)。即便如此，**關鍵的參數更新流程已打通**：經驗→訓練→模型更新→保存，這意味著BioNeuron決策核心並非靜態模板，而是可以透過經驗學習改變。

**結論：** 在最新版本中，BioNeuron 的模型參數確實能透過訓練過程得到更新，從而改變其推理輸出。相關模組（ModelUpdater/Trainer）已實現將經驗數據用於訓練並更新 ScalableBioNet 權重[[29]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L80-L88)[[33]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L109-L117)。因此只要運行訓練流程，BioNeuron 未來產生的計畫和決策有望逐步改進。需要後續開發的是自動觸發訓練及更高級的學習策略，但整體閉環中的「**執行經驗反哺模型**」功能已具備基本可行性。

## 目前功能完備度總結

歸納而言，AIVA 最新版本已經將**AI 閉環的主要環節串接**起來，但**部分細節尚未完成**，導致整體系統目前更多處於模擬驗證階段，可供**實作練習**的部分如下：

* **已實作/可運行的部分：** 攻擊計畫的任務分解與基本執行流程、執行過程的詳細追蹤記錄、AST 與執行結果對比分析、經驗資料的保存，以及基於經驗的模型訓練更新[[38]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L119-L127)[[20]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L480-L489)[[29]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L80-L88)。換言之，開發者可以利用現有框架模擬從計畫→執行→得到經驗→訓練模型的循環，觀察各模組的互動和數據流。
* **尚未完成/需後續開發的部分：**
* **攻擊計畫自動生成：** 目前仍使用固定範本，AI 尚不能根據任意使用者需求生成全新計畫[[1]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L107-L115)。後續需實現 BioNeuron 真正的計畫生成功能（可能結合LLM或強化學習策略）。
* **真實工具整合執行：** 現在所有執行結果皆為模擬，缺乏實際漏洞掃描和攻擊工具的對接[[6]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L382-L390)[[7]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L128-L136)。需實作 RabbitMQ 結果接收機制及各 Function 模組的任務執行，以達到真正「動手」攻擊目標的效果。
* **自動學習閉環：** 雖有經驗庫和訓練模組，但訓練的觸發和強化學習細節（如獎勵計算）仍未完善[[24]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L183-L189)[[36]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/learning/model_trainer.py#L140-L149)。未來應加入自動判斷何時從經驗啟動模型更新，以及更精細的強化學習迴圈。

總體來看，AIVA 的**AI 閉環架構已基本成形**：從使用者輸入，到AI決策攻擊步驟，經由模擬執行獲得結果，再將結果存為經驗、用於模型訓練，形成反饋閉環的各個模組都已有初步實現[[19]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L476-L485)[[21]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L481-L489)[[33]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L109-L117)。但目前有部分關鍵環節停留在樣板或 TODO 狀態。因此**現階段可以進行閉環「演練」的多為模擬實驗**；真正的自動駭客攻防閉環還需待計畫生成和真實執行的功能完善後才能完全實現。

[[1]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L107-L115) [[2]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L117-L125) [[24]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L183-L189) [[38]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py#L119-L127) training\_orchestrator.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/training/training_orchestrator.py>

[[3]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/core/aiva_core/bio_neuron_master.py#L331-L339) bio\_neuron\_master.py

<https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/core/aiva_core/bio_neuron_master.py>

[[4]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L99-L107) [[5]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L119-L127) [[6]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L382-L390) [[13]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py#L233-L241) plan\_executor.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution/plan_executor.py>

[[7]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L128-L136) [[8]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L160-L169) [[9]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L200-L208) [[10]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L128-L135) [[14]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py#L166-L174) task\_executor.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/task_executor.py>

[[11]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L114-L123) [[12]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L150-L158) [[15]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L248-L257) [[16]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py#L259-L267) execution\_monitor.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/execution_tracer/execution_monitor.py>

[[17]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/analysis/ast_trace_comparator.py#L98-L106) [[18]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/analysis/ast_trace_comparator.py#L124-L132) ast\_trace\_comparator.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/analysis/ast_trace_comparator.py>

[[19]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L476-L485) [[20]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L480-L489) [[21]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L481-L489) [[27]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L530-L538) [[28]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py#L539-L546) bio\_neuron\_core.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/bio_neuron_core.py>

[[22]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L82-L91) [[23]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L98-L106) [[25]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L138-L147) [[26]](https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py#L160-L169) experience\_repository.py

<https://github.com/kyle0527/AIVA/blob/fa2edd25288f2cb61b62149c80df80e4c797b2b4/services/integration/aiva_integration/reception/experience_repository.py>

[[29]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L80-L88) [[33]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L109-L117) [[34]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L84-L93) [[35]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py#L189-L198) model\_updater.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/model_updater.py>

[[30]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/data_loader.py#L48-L56) [[31]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/data_loader.py#L80-L88) data\_loader.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/data_loader.py>

[[32]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/trainer.py#L218-L225) trainer.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/ai_engine/training/trainer.py>

[[36]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/learning/model_trainer.py#L140-L149) [[37]](https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/learning/model_trainer.py#L156-L165) model\_trainer.py

<https://github.com/kyle0527/AIVA/blob/848399d1f7cedd35f92e0a86efd1c1d3008f7b73/services/core/aiva_core/learning/model_trainer.py>