與學長整體系統的差異: 我**只使用預測模組，沒用到改道**的部分。因此整體程式架構與學長是接近的只是刪除了改道的部分。建議**先跑一次學長的**code(<https://github.com/wmnet741/Vehicle-Rerouting-Algorithm/tree/main/MuKun-2022/Code/Algorithms> )，請先閱讀**DTPDVR\_process的說明**，有不懂的再詢問。

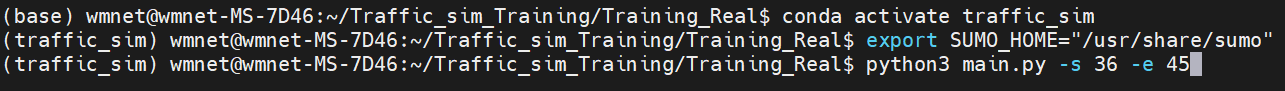
程式整體架構:  
Training: 完成**數據收集**與LSTM預測**模型的訓練**

Testing: 利用Training的模型結合Driver Behavior Model進行**模型表現評估**。

(參考資料: reference/Routing)

程式說明:

**Training\_Real:**



* 模擬前記得確認使用正確的od\_route\_file.odtripsXX.rou放到data/01，以及tripinfo.xml到Experiment\_result
* run main.py ->產生 bd, txt\_result (已打包好完整一次模擬的程序，其他單一檔案可用於測試個別檔案結果輸出)
* run txt\_to\_csv.py ->產生 csv\_result
* run bd\_txt\_to\_csv.py ->產生 bd\_csv\_result
* run vehavg\_inst.py ->產生 interavg\_result, interZ\_result
* run finalavg.py ->產生 finalavg\_result, finalZ\_result
* run train.py ->產生 models 資料夾

一張含有 文字, 螢幕擷取畫面, 字型, 設計 的圖片

自動產生的描述

**Attack Methodology:**

* 模擬前記得確認使用正確的od\_route\_file.odtripsXX.rou放到data/01，以及tripinfo.xml到Experiment\_result
* 複製**Training\_Real**中的txt\_result
* run main.py ->完整一次模擬的程序，其他單一檔案可用於測試個別檔案結果輸出
* run txt\_to\_csv.py ->進行攻擊並產生 csv\_result
* run vehavg\_inst.py ->產生 interavg\_result, interZ\_result
* run finalavg.py ->產生 finalavg\_result, finalZ\_result
* run train.py ->產生 models 資料夾

一張含有 文字, 螢幕擷取畫面, 網頁, 軟體 的圖片

自動產生的描述

**Defense Methodology:**

* 模擬前記得確認使用正確的od\_route\_file.odtripsXX.rou放到data/01，以及tripinfo.xml到Experiment\_result
* 複製**Training\_Real**中的csv\_result並更名為**real\_csv\_result**及 **Attack Methodology** 中的**csv\_result**
* run main.py ->完整一次防禦模擬的程序，其他單一檔案可用於測試個別檔案結果輸出
* run concat.py ->real\_csv\_result與csv\_result合併產生concat資料夾
* run defense.py ->進行竄改數據識別產生defense資料夾
* run combine\_defense.py ->將上述產出的檔案中根據**路段**組成單一檔案以利後續學習資料間關係式，產生SVM\_ref資料夾
* run SVM.py -> 繪製SVM並產生SVM資料夾與Reference/relationships\_defense檔案
* run Linear\_Regression.py ->繪製Linear\_Regression並儲存於Reference資料夾，同時產生Reference/relationships\_recover檔案
* run recover.py -> 進行數據修復並產生 Sterilized資料夾
* run vehavg\_inst.py ->產生 interavg\_result, interZ\_result
* run finalavg.py ->產生 finalavg\_result, finalZ\_result
* run train.py ->產生 models 資料夾
* run XXX\_withSVM.py -> 使用Reference/relationships\_defense與Reference/relationships\_recover資料完成防禦與修復並產生Sterilized資料夾

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

**Testing:**

一張含有 文字, 字型, 數字, 螢幕擷取畫面 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型, 圖表 的圖片

自動產生的描述

1. 複製**Training\_Real生成的bd\_csv\_result, csv\_result, interavg\_result**
2. 複製 **Training\_Attacks生成的**finalavg\_result 重新命名為 mean , finalZ\_result 重新命名為 meanZ, models
3. Run driver.py ->driver
4. **將driver 複製到實驗數據/RMSE\_experiment並更名為esti\_X, run Analyze.ipynb，到Result收結果。**

**工具: SUMO**，進行交通流量模擬，先了解基礎使用

Tool & Scenario: 建立情境所需的工具與部分跑完的結果

* 執行前都先看一下 ReadMe.docx

Scenario 只是儲存各情境下的odtripsX.rou.xml，主要看tool

* InductionLoop/MakedDet ->建設路邊sensor收資料
* MakeScenario(詳細情境怎麼設計可以看睦昆的論文)

收數據: 140.114.77.84 -> 目前model training有問題

運行環境: 140.114.77.72，anaconda 中 traffic\_sim 環境

一張含有 螢幕擷取畫面, 文字, 字型 的圖片

自動產生的描述

運行方式: 開5個tmux 在 server. tmux 的操作可以上網查很簡單。



宣告SUMO\_HOME



執行:

python3 main.py -s 0 -e 9

python3 main.py -s 9 -e 18

python3 main.py -s 18 -e 27

python3 main.py -s 27 -e 36

python3 main.py -s 36 -e 45

目前環境有被破壞，建議之後重建環境或是debug修復環境

File解釋:

main.py ->產生 bd ,txt\_result

bd: 紀錄車輛ID 、車輛碰觸中間sensor的速度、driver behavior 參數

txt\_result: 紀錄車輛ID、車輛碰觸前面sensor的時間、車輛碰觸中間sensor的時間、車輛碰觸中間sensor的速度、車輛碰觸後方sensor的時間

txt\_to\_csv.py ->產生 csv\_result

csv\_result: 紀錄車輛ID(vehid)、車輛碰觸前面sensor的時間(entry)、 車輛碰觸中間sensor的時間(mid)、 車輛碰觸中間sensor的速度(instspeed)、 車輛碰觸後方sensor的時間(exit)、道路長度(length)

run bd\_txt\_to\_csv.py ->產生 bd\_csv\_result

bd\_csv\_result: 紀錄車輛ID(vehid) 、車輛碰觸中間sensor的速度(instspeed)、driver behavior 參數(Bd)

run vehavg\_inst.py ->產生 interavg\_result, interZ\_result

interavg\_result: 紀錄每次實驗各個道路每5分鐘平均/瞬時速度: interval(5分鐘一個間隔)、instspeed(每5分鐘的平均瞬時速度)、avgspeed(每5分鐘的平均速度)

interZ\_result: 紀錄每次實驗各個道路每5分鐘的Zmax、Zmin、qvalue

run finalavg.py ->產生 finalavg\_result, finalZ\_result

finalavg\_result: 平均前五次實驗的interavg\_result/out0~4的file，當作 mean

finalZ\_result: 平均前五次實驗的interavg\_result/out0~4的file，當作 meanZ

run train.py ->產生 models 資料夾，使用interavg\_result/out5~outx的所有file 減去 finalavg\_result(mean)來訓練模型

run driver.py ->產生 drivers, Timer, log 資料夾

driver: 運用driver behavior model轉換後的數據及相關參數資料，當作 RMSE 衡量的esti

Timer, log: log 資料