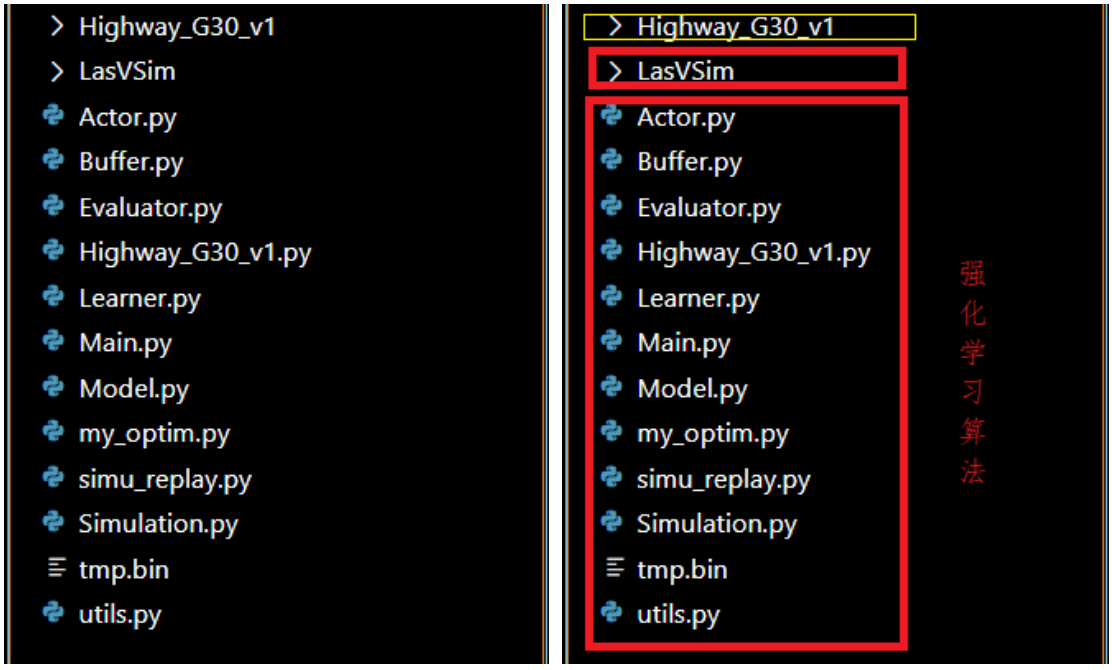
# 长安项目代码简介和使用说明

为项目组成员快速了解代码架构和使用，以多车道代码为例，简要介绍代码结构和实现逻辑。算法具体实现过程和细节，可通过调试学习。

## 1. 代码架构

由下图，代码包括强化学习算法模块，LasVSim 仿真模块，和结果存储部分




## 2. 强化学习算法

（1）**Main.py** 实现功能是设定算法相关参数，训练任务相关参数，以及并行训练相关参数；构建学习器，执行器，评估器，经验池；实现训练和测试功能。运行模式有两个，一个是训练模式“train”，一个时测试模式”simu”。训练模式下，算法会以固定的训练间隔，存储训练得到的值网络和策略网络到 Highway_G30_v1 文件夹下面。测试模式下，算法会调用得到的策略网络进行测试。

（2）**Actor.py** 实现功能是基于本地策略网络和环境进行交互以采集数据，并将数据随机发送给到经验池。

（3）**Buffer.py** 实现功能是构建经验池将获得的数据进行存储，并在内部进行采样，随机发送给学习器。

（4）**Learner.py** 实现功能是依托接收到的数据，根据本地的值网络和策略网络计算更新梯度，并利用计算得到的梯度对共享网络进行更新。

（5）**Evaluator.py** 实现功能是对得到的策略网络和值网络进行评估，得到网络的性能并存储到相应的性能文件中。

（6）**Model.py** 实现功能是策略网络，编码网络，值网络的构建。

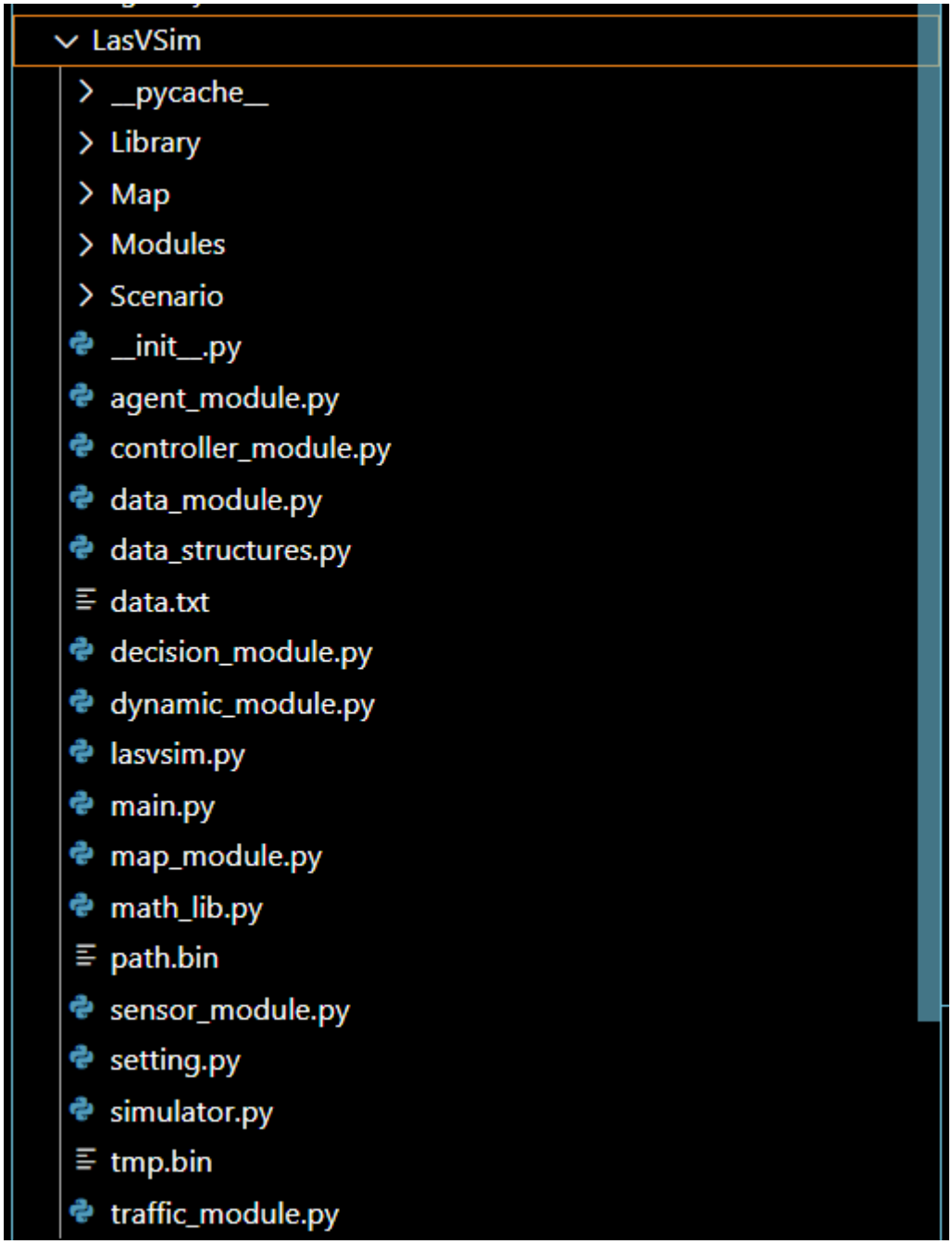
（7）**my_optim.py** 实现功能是利用 Adam 优化器计算网络的损失。

（8）**Highway_G30_v1.py** 实现功能是自车与环境的交互，自车、周车、环境信息的获取，以及不同情况下的回报函数设定等。

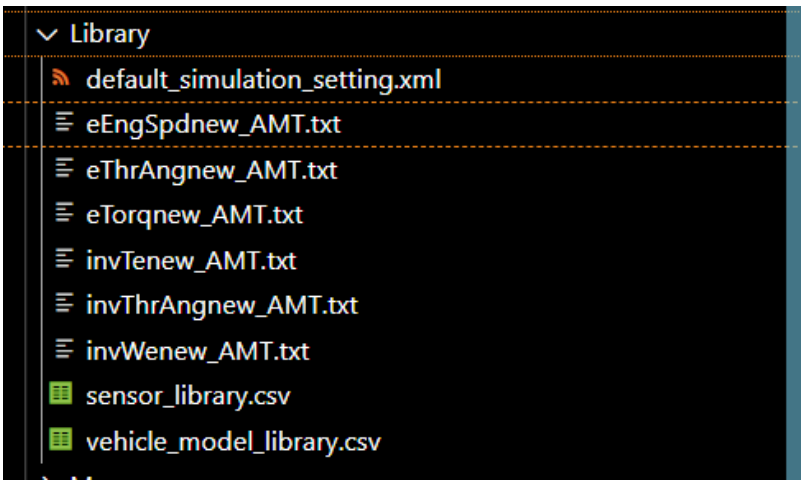
（9）**Simulation.py** 用于测试策略网络的性能，并将测试过程中的相关数据，如自车行驶状态，控制量等，存储到 **Highway_G30_v1** 文件夹下面的 **test** 文件夹下面。

（10）**simu_replay.py** 用于统计测试结果，比如碰撞次数，驶出车道的次数等。

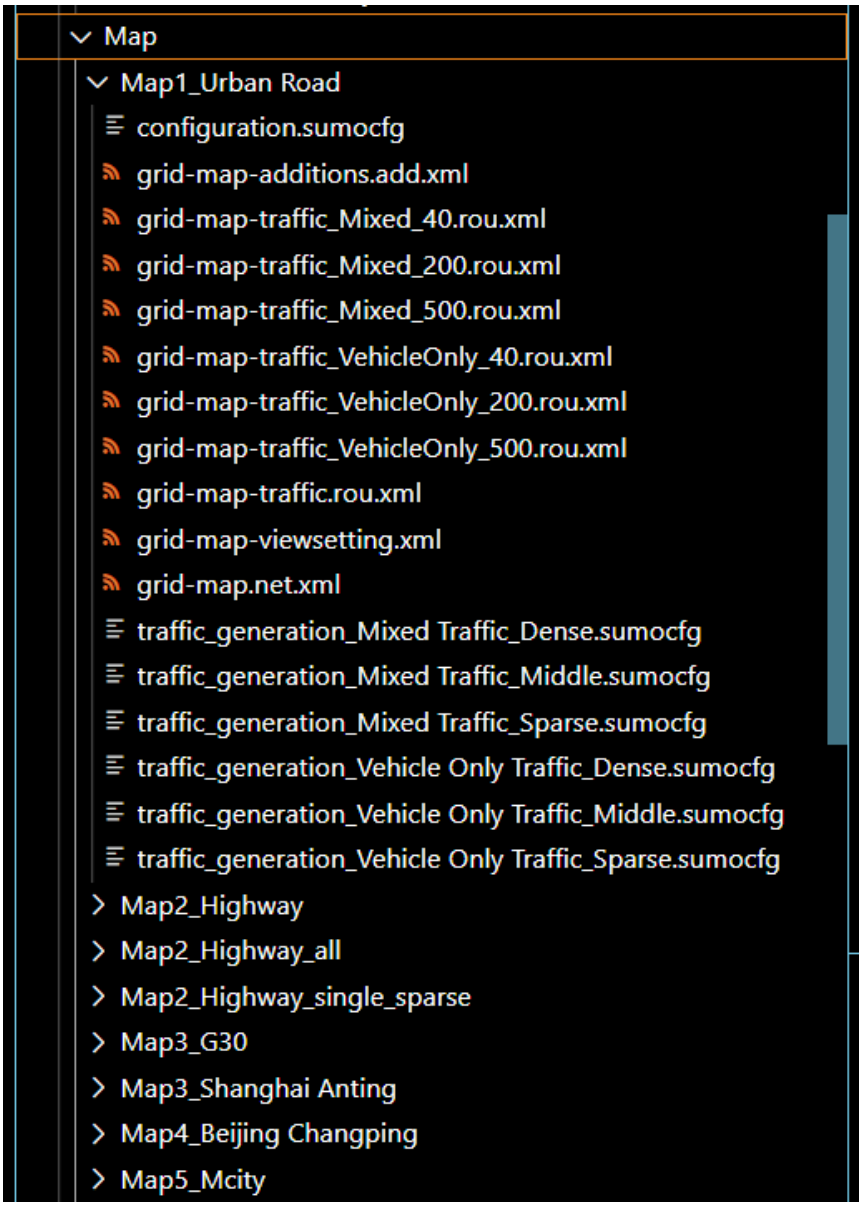
3. **LasVSim 仿真模块**



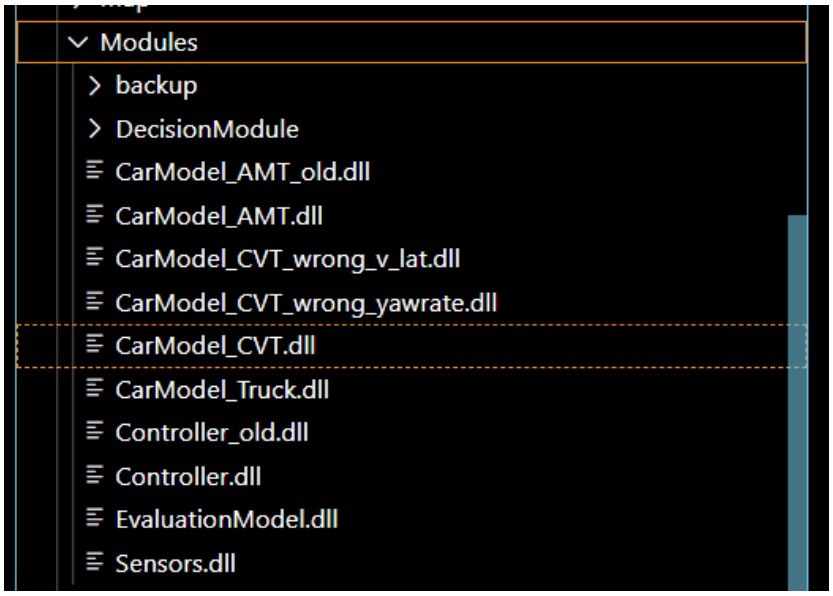

（1）**Library 文件夹**下一些车辆相关的参数文件



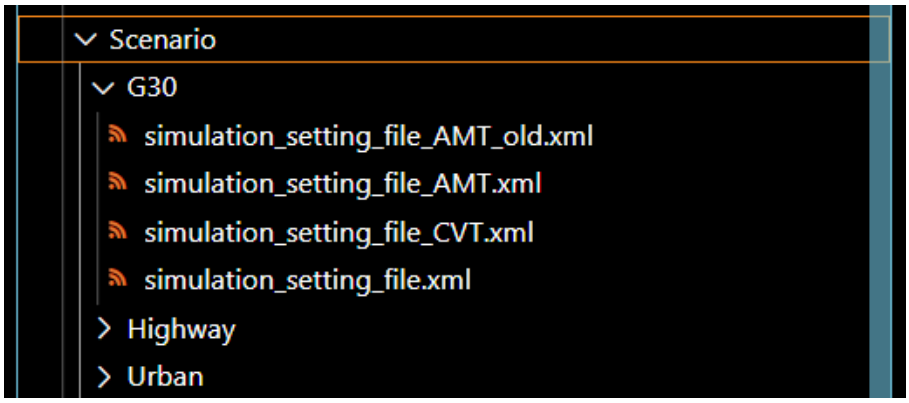

（2）**Map 文件夹**是一些我们公开的地图，每个地图包含最少包含三个文件，一个地图道路结构.net.xml 文件，一个地图交通流配置的.rou.xml 文件，一个仿真运行文件.sumocfg。




（3）**Modules 文件夹**下是一些整车和传动系统模型文件，在自车控制过程中会调用：




（4）**Scenario 文件夹**下是不同工况下的仿真配置文件




（5）**agent_module.py** 是自车模块，包括路径选择，感知信息获取，自车状态更新等功能。

（6）**controller_module.py** 是跟踪控制模块，调用车辆动力学模型，利用 PID 控制进行跟踪控制。

（7）**data_module.py** 模块是一些自车相关数据处理和获取的过程。

（8）**data_structures.py** 模块分别定义了自车，周车，道路的数据结构类。

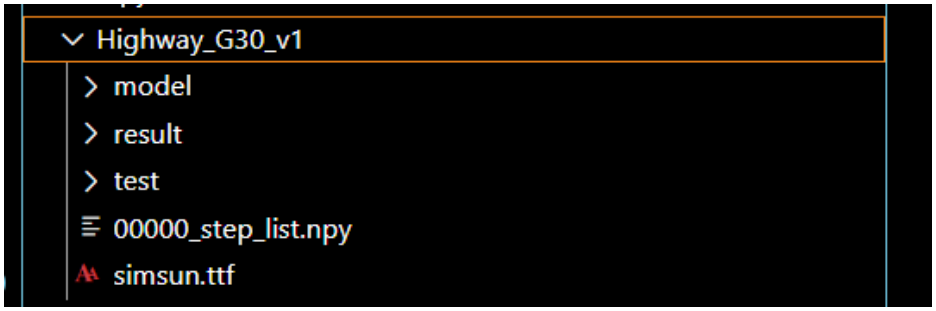
（9）**decision_module.py** 模块调用决策模型对车辆期望轨迹进行规划。

（10）**dynamic_module.py** 模块调用车辆动力学模型，更新车辆状态。

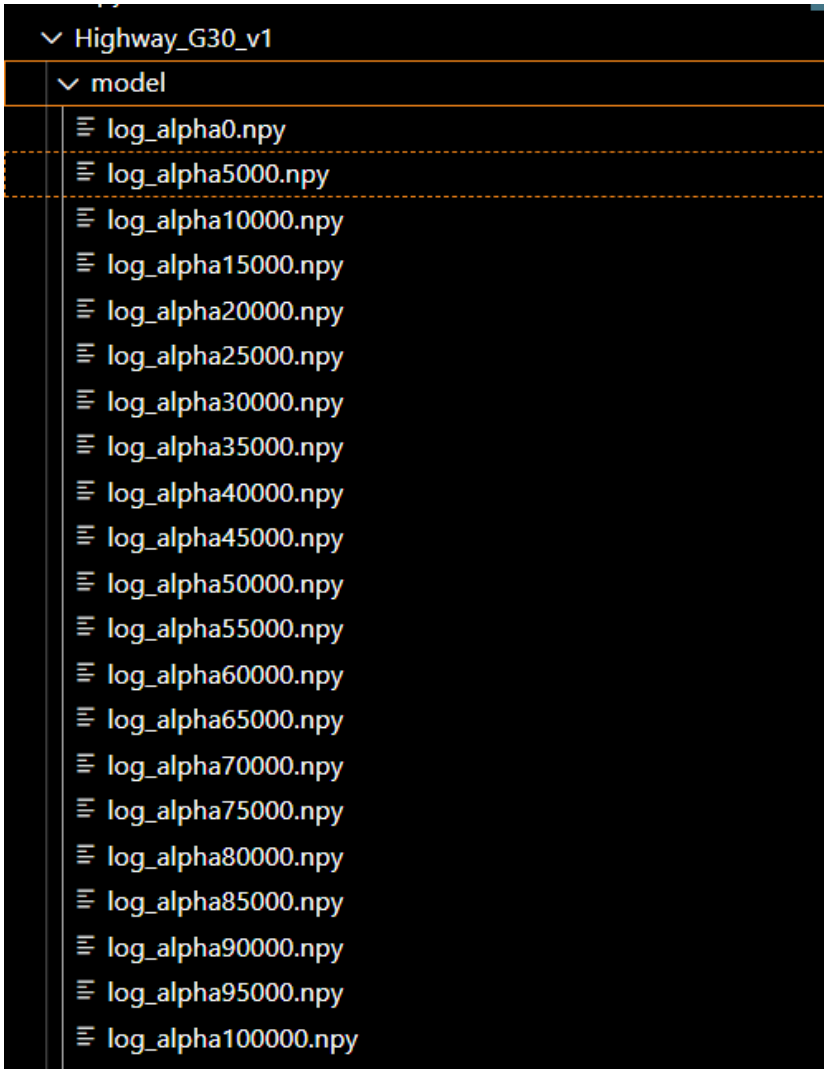
（11）**map_modlue.py** 模块是根据自车当前位置，获取导航信息。

（12）**lasvisim.py** 模块是整个 lasvsim 初始化，运行过程，和结束相关的一些函数

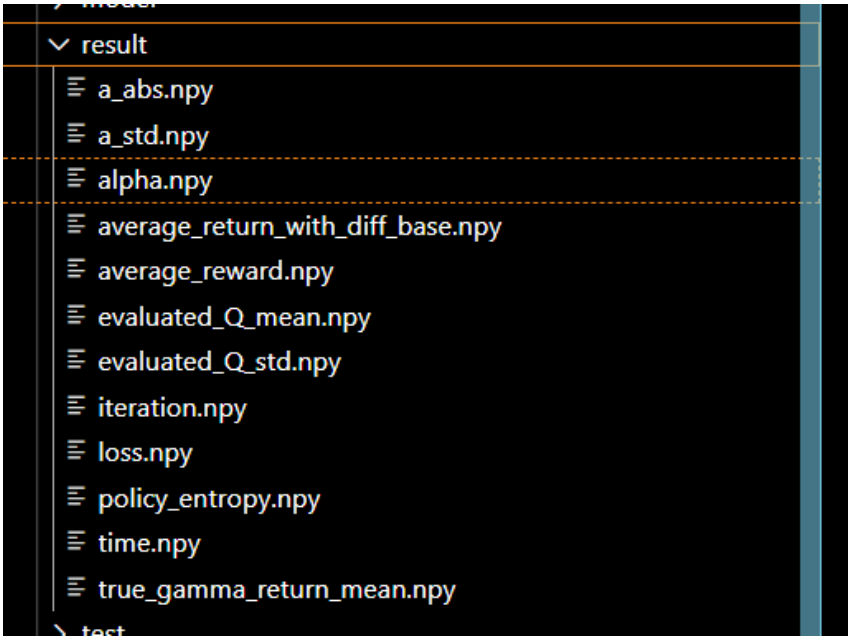
**4. 训练结果存储部分**



（1）**Model 文件夹**下存储的是训练过程中每个固定步数生成的策略网络和值网络。




（2）**result 文件夹**下存储的是训练过程中，网络的性能表现。



（3）**test 文件夹**存储的是网络测试时，每一次测试对应的环境，车辆，道路数据。