

End-to-end Driving via Conditional Imitation Learning 复现记录

杨安东

目录

1	说明	2
2	实验环境	2
2.1	部署设计	2
2.2	训练数据收集	2
3	神经网络实现	3
3.1	网络结构	3
3.2	参数	3
4	代码结构设计	4
5	结果分析	4
6	想法记录	4
7	进度记录	5
7.1	第一周	5
7.2	第二周	5
7.3	第三周	5
7.4	第四周计划	5

1 说明

本报告将介绍实验的整体设计并在最后记录每周进度与下一周计划。

2 实验环境

2.1 部署设计

实验使用 CARLA 模拟器，其分为客户端服务端两部分，部署设计框架如图 1所示。在训练阶段只需要数据集即可，网络模型不需要与 CARLA 进行交互。模型测试阶段网络模型与 CARLA 客户端交互获取图像并发送控制命令。之后通过日志记录进行本地可视化。

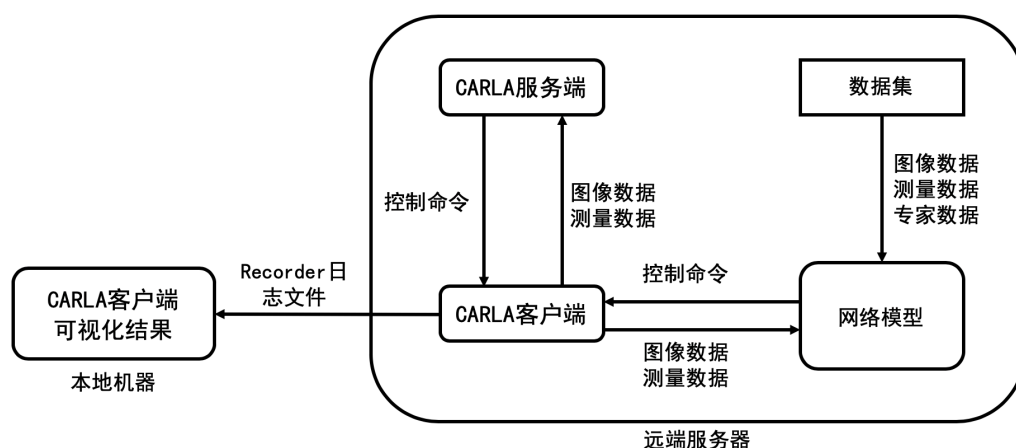


图1 实验设计框架

2.2 训练数据收集

原始训练数据分为一幕一幕收集，包含图像数据，测量数据即车辆速度（是可选项，若在实际车辆中噪声较大则不用），专家控制命令（转向角和加速度）三部分。每一幕从不同的位置开始，提供目的地，让专家控制车辆前往目的地。

图像数据 (200*88) 从车辆前方的三个摄像头获取，左右两个摄像头的的数据作为错误数据，中间摄像头数据作为正常数据。每帧图像附带对应的专家的控制命令，为前文提到的错误数据提供使得车辆恢复到正常路线的控制命令。为了增强鲁棒性，偶尔在专家控制中叠加扰动，让专家从扰动中恢复，控制命令只记录专家采取的命令，不记录叠加命令。如图 2所示。

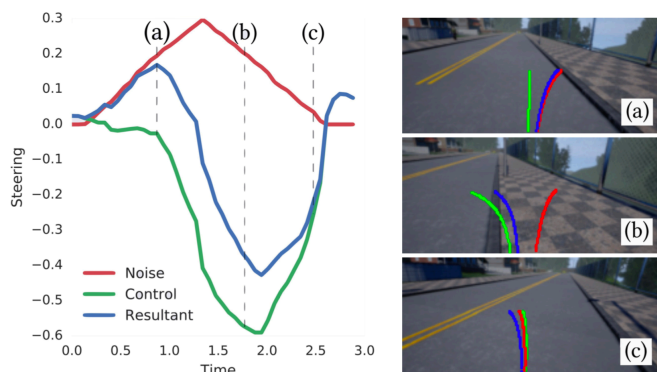


图2 添加控制命令扰动

对于采集到的原始数据进行数据增强，从一系列变换中选取一部分对图像进行处理。变换包括：

1. 亮度，对比度变化
2. 扭曲畸变
3. 高斯模糊，高斯噪声，椒盐噪声
4. 区域丢弃（随机选取一定数量的长方形丢弃，每个长方形占全图面积 1%）
5. 不涉及几何变换例如旋转，对称等

总的来说数据集包含：图像与对应车辆速度、专家转向角、专家油门、专家刹车、控制命令（沿车道行驶、左转、右转、直行）。

实验计划使用论文给出的收集好的训练数据集，大小为 24G，链接如下：

<https://github.com/carla-simulator/imitation-learning/blob/master/README.md>

3 神经网络实现

神经网络将图像、车速（可选）、控制命令作为网络输入。输出为转向角、油门、刹车、预测车速。

3.1 网络结构

网络输出包含转向角，油门，刹车，预测速度，四个部分。

论文中所示的网络结构在最后具有一个选择类型的结构，为了训练方便，每次都计算 4 个分支网络，之后根据控制命令添加掩码将对应分支之外的网络输出变为 0，从而使得反向传导时不影响权重，只训练控制命令对应的分支。

网络结构如图 3所示：

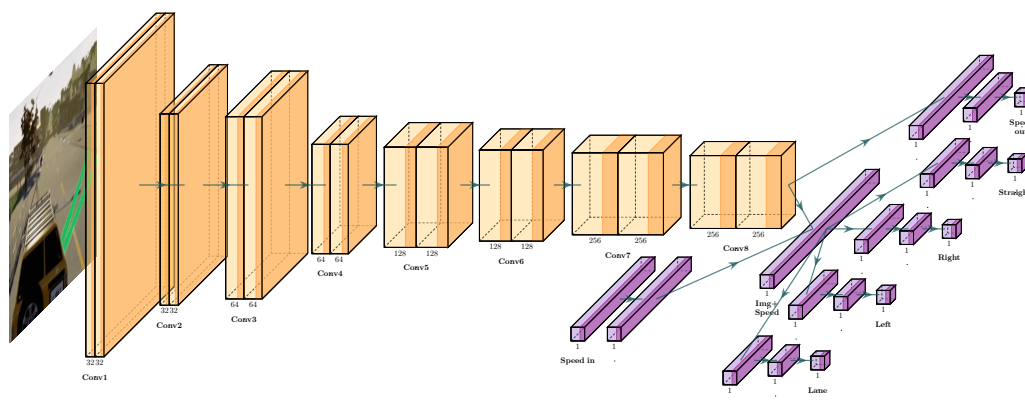


图3 神经网络结构

3.2 参数

1. 损失函数使用均方损失 `nn.MSELoss()`
2. 优化器使用 `adam`，学习率 0.0002
3. `batch = 4` 每个控制命令的 `batch` 数目一样
4. 训练轮数由实际训练效果决定
5. `batch_size` 和 `DataLoader` 的 `num_workers` 由处理器核心数决定

4 代码结构设计

代码整体设计如图 4 所示。数据读取数据读取主要使用 `torch.utils.data.DataLoader`，其需要 `Dataset` 类型，`torch.utils.data.Dataset` 是抽象类，通过继承 `Dataset` 类并重写 `__len__` 与 `__getitem__` 方法来实现自定义数据读取。数据读取后 `main` 函数读取网络进行实例化，之后通过 `train` 函数进行训练，完成一定次数训练后通过 `output_log` 函数输出日志文件。将日志文件下载到本地后可可视化观察训练效果。同时通过将参数单独设置为一个类，统一参数设置数据。

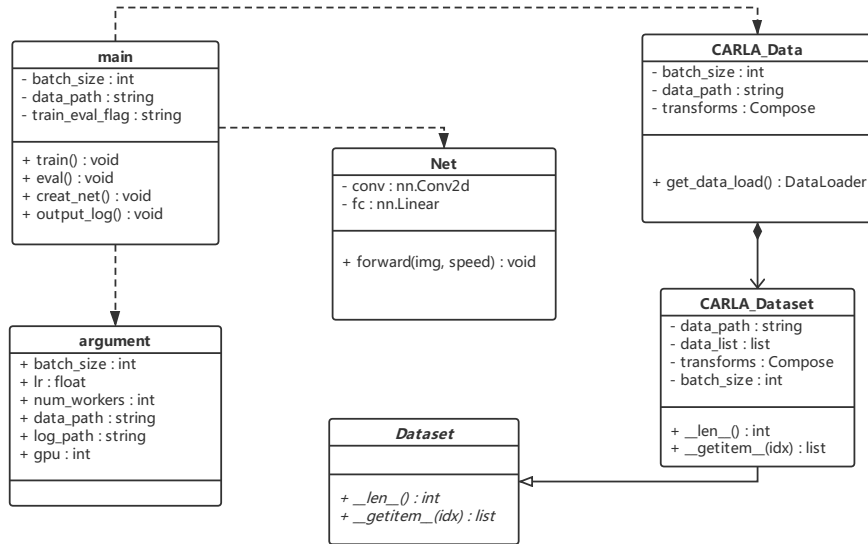


图4 代码 UML 类图

5 结果分析

指定一个起点和终点，记录车辆从起点到终点错过的转弯数，需要人为干预的次数和用时来衡量方法有效性。不断重复此过程，取平均获得最终评价结果。同时将测试过程可视化。

使用 `driving-benchmarks` 实现测试部分。链接：<https://github.com/carla-simulator/driving-benchmarks>，其提供一个测试函数 `run_driving_benchmark()`，主要参数包括：

1. `agent`：根据环境输入作出决策
2. `experiment_suites`：指定起点与终点，点在 `carla` 环境中已经定义
3. `carla` 服务端的 IP 与端口

通过调用 `run_driving_benchmark`，即可根据指定的起点终点，测试 `agent` 的表现，结果会以统计数据形式输出，包含：

1. 任务完成概率
2. 完成任务平均路程
3. 各类碰撞次数
4. 冲出道路次数

6 想法记录

1. 考虑过路口的时候加入红绿灯信号的影响。
2. 考虑其他车辆转向灯的影响。

7 进度记录

github 仓库: <https://github.com/andongyang/Re-implement>

7.1 第一周

1. 完成实验方案设计, 确定代码结构、神经网络结构。
2. 编写神经网络部分代码, 编写部分数据读取部分代码编写。
3. 学习 CARLA 使用方法。
4. 搭建 CARLA 环境。

7.2 第二周

1. 学习 CARLA 提供的 benchmarks-CoRL2017
2. 完成训练部分代码编写
3. 测试 h5 文件读取部分功能

7.3 第三周

1. 测试 CARLA 日志文件还原场景。
2. 借用 CoRL2017 完成测试部分代码编写。
3. 完成模型训练, 获得一个初步的训练结果。

7.4 第四周计划

1. 解决连接服务端 timeout 问题。
2. 测试获得的训练结果并根据反馈进行参数调整。
3. 获得最终训练结果。