CARLA中端到端模型的训练与闭环评测

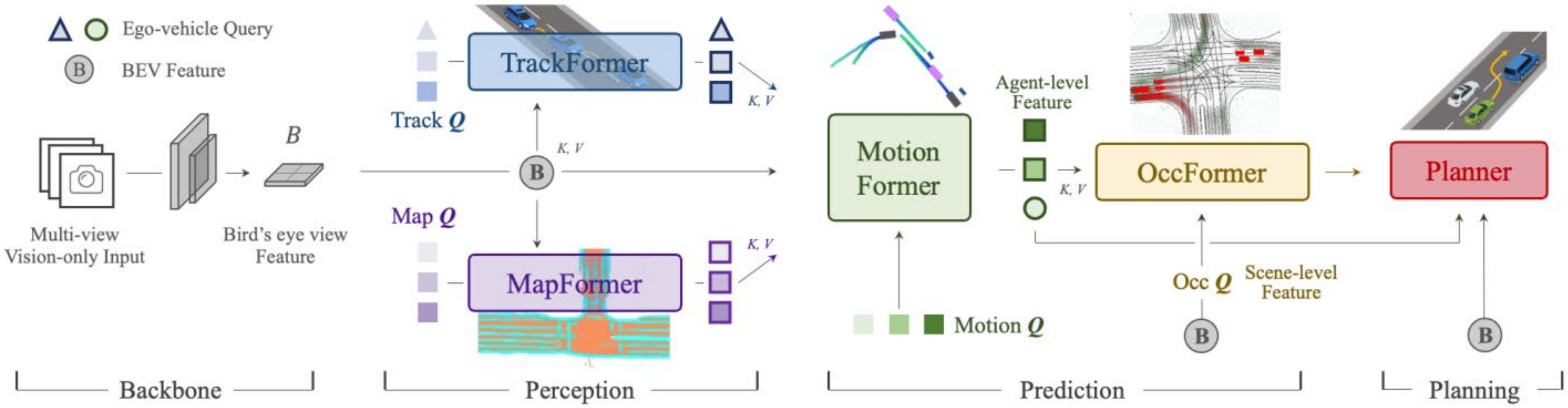
贾萧松

2024年6月10日

端到端自动驾驶

- 端到端自动驾驶：

  以传感器为输入，最终输出目标是自车规划轨迹或者底层控制信号



经典端到端自动驾驶模型：UniAD

- 端到端模型的评测 -> 面向最终驾驶结果的评测 （让车开起来！）

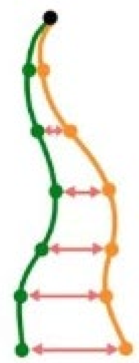
Planning-oriented Autonomous Driving, CVPR2023 Best Paper

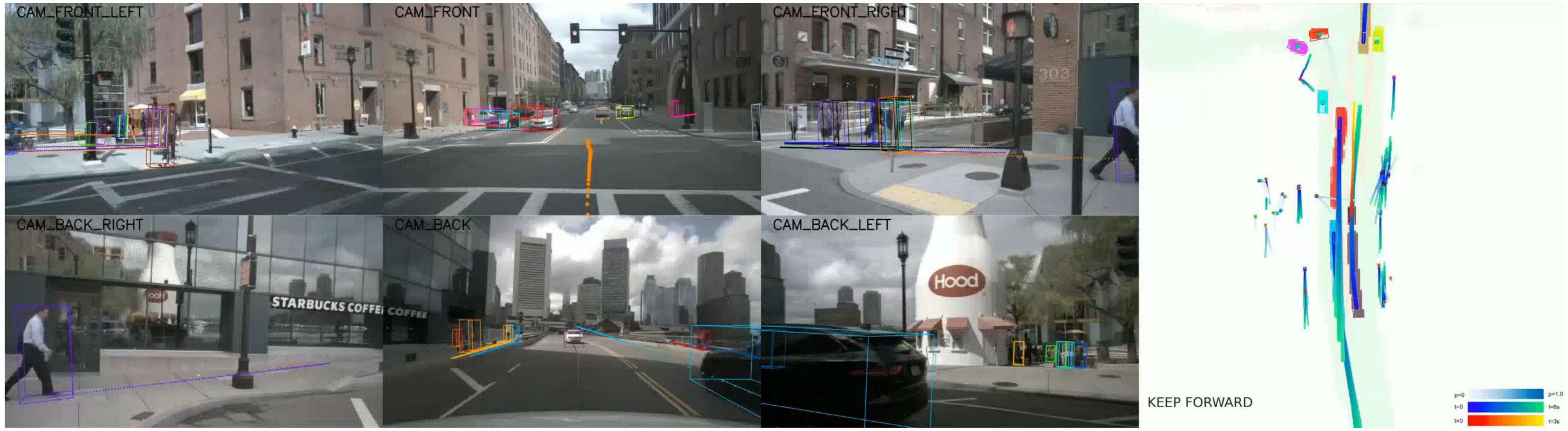
端到端自动驾驶的评测方式

- 两大类评测方式：
  1. 开环评测：在预先采集好的数据集上，逐帧对比模型预测的自车未来动作/轨迹与采集时的记录下来的值。

     指标 – L2误差：预测的自车未来轨迹与真实未来轨迹的每个时间步欧几里得距离的平均

     指标 – 碰撞率：假设他车不受自车影响，预测的自车轨迹与未来时刻他车3D框碰撞的比例。





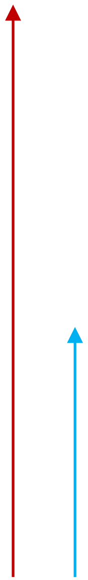
UniAD可视化，逐帧画出各模块结果，并不真实开车

端到端自动驾驶的评测方式

- 两大类评测方式：
  1. 开环评测：

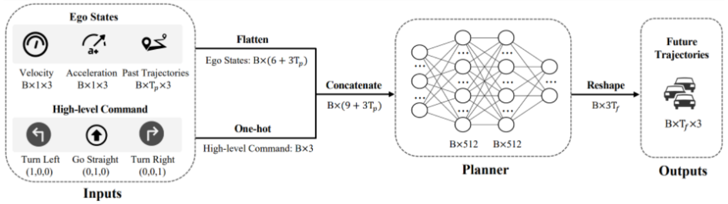
     Pros：可以使用真实世界的感知数据集，难度高多样性大

     Cons：自动驾驶长尾分布的特性，会导致没有经过专门平衡的数据集（例如nuScenes）中大多数轨迹都是简单直行。直行时的L2误差集中在径向，实际应用中对驾驶安危影响较小。缺乏交互式场景也导致数据集可以用类匀速直线运动拟合。例如，AD-MLP中，无需视觉输入/激光雷达输入（盲人开车），仅凭过去帧的速度、加速度即可取得SOTA L2误差。



高L2误差的轨迹

低L2误差仍然出现危险的场景



AD-MLP架构

Rethinking the Open-Loop Evaluation of End-to-End Autonomous Driving in nuScenes, arXiv 2305.10430

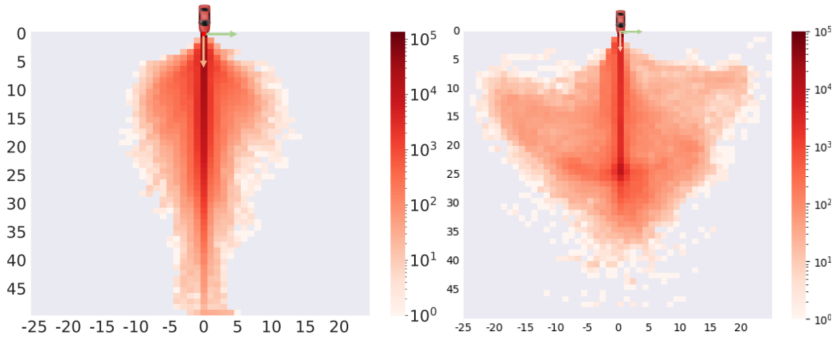
Is Ego Status All You Need for Open-Loop End-to-End Autonomous Driving? CVPR 2024

端到端自动驾驶的评测方式

- 两大类评测方式：
  1. 开环评测：

     那么在均衡后的数据集上，开环指标会有更大的意义吗？

     有，但不多。可以用来判断模型是否收敛（AD-MLP无法取得低L2误差），但**对判断驾驶能力基本上没有相关性**



(a) **nuScenes** [2] dataset drawn by [33]. (b) Bench2Drive Dataset

Figure 6: **Distribution of ego vehicle's future location.** Bench2Drive possesses more turning trajectories, indicating better action diversity and thus providing better training data and having less gap between open-loop and closed-loop evaluation.

Bench2Drive数据集上自车未来轨迹的分布

Table 3: **Open-loop and Closed-loop Results of E2E-AD Methods in Bench2Drive**. Avg. L2 is averaged over the predictions in 2 seconds under 2Hz. * denotes expert feature distillation.

Method	Input	Open-loop Metric	Closed-loop Metric	
		Avg. L2 (meter) ↓	Driving Score ↑	Success Rate(%) ↑
AD-MLP [51]	Ego State	3.64	9.14	0.00
UniAD-Tiny [17]	Ego State + 6 Camera	0.80	32.00	9.54
UniAD-Base [17]	Ego State + 6 Camera	**0.73**	37.72	9.54
VAD [27]	Ego State + 6 Camera	0.91	**39.42**	**10.00**
TCP* [47]	Ego State + Front Cameras	1.70	23.63	7.72
TCP-ctrl*	Ego State + Front Cameras	-	18.63	5.45
TCP-traj*	Ego State + Front Cameras	1.70	36.78	26.82
ThinkTwice* [25]	Ego State + 6 Cameras	**0.95**	39.88	28.14
DriveAdapter*([23]	Ego State + 6 Cameras	1.01	**42.91**	**30.71**

Bench2Drive数据集上开环与闭环结果

Is Ego Status All You Need for Open-Loop End-to-End Autonomous Driving? CVPR 2024
Bench2Drive: Towards Multi-Ability Benchmarking of Closed-Loop End-To-End Autonomous Driving. arXiv:2406.03877

端到端自动驾驶的评测方式

- 两大类评测方式：

  2. 闭环评测：给定起点终点，让模型开车，根据安全性、高效性、舒适性等打分。

     Pros：与真实应用时的效果相关性更高

     Cons:

     a) 想使用真实世界渲染，采用nuplan/NAVSIM（CVPR 2024 比赛），那么需要假设自车动作对未来渲染不产生影响
     b) 使用CARLA等仿真平台，但渲染距离现实Gap较大
     c) （未来方向）使用GenAD/Vista等提供渲染

端到端自动驾驶闭环评测 - CARLA

- CARLA是一款基于Unreal的开源仿真平台，是自动驾驶社区使用最广泛的仿真器。
- 其主体为仿真器，支持从底层的添加新素材（新相机模型、新城镇、新车辆）到验证单车车辆驾驶能力，再到交通模拟/V2X，包罗万象，DIY程度较高。

- **端到端自动驾驶方法在CARLA中的评测流程（以模仿学习为例）：**
  1. 采集数据：CARLA官方并不提供驾驶数据，研究者需要自行使用CARLA采集用于训练模型的数据
  2. 模型训练：使用自己的训练代码库(mmcv,Bench2Drive,pytorch lighting, etc)，Dataset类对接采集到数据格式，完成模型训练
  3. 模型评测：使用CARLA的评测代码库，在CARLA中真实驾驶

CARLA: An Open Urban Driving Simulator. arXiv 1711.03938

端到端自动驾驶闭环评测 – 数据采集

- CARLA官方仅提供仿真及评测功能，没有官方数据集
- TODO：
  - 制定采集计划：确定采集数据的城市、路线、事件、天气
  - 定义数据内容与格式：调用CARLA官方API，自定义每时每刻需要存储的数据，例如：传感器（相机/LiDAR/Radar），标注（3D框，车道线）
  - 设计专家模型 (or 手动开)：在能够获得仿真中所有信息的条件下，设计算法能跟完成给定的路线驾驶
  - （可选）并行启动脚本：并行启动多个CARLA，在多GPU上完成采集
  - 后处理：剔除掉有违规的路线（错误样本影响模仿学习性能）以及渲染出错的样本

Figure 7: **Bugged Rendering of CARLA.**

端到端自动驾驶闭环评测 – 数据采集

- 数据采集 - 制定采集计划：
  - CARLA中驾驶路线都是yaml的形式，需要用户自行制定：
    - □ 起点终点（中间的路点可以写代码调用CARLA API得到）
    - □ 天气与光照状况（晴天、雨天、阴天、白天、晚上）
    - □ 发生的事件（scenario）

```
<routes><route id="1711" town="Town12">
    <waypoints>
      <position x="-497.6" y="3672.9" z="364.9" />
      <position x="-497.6" y="3670.9" z="364.8" />
      <position x="-497.6" y="3668.9" z="364.8" />
      <position x="-497.6" y="3666.9" z="364.8" />
      <position x="-497.6" y="3664.9" z="364.8" />
      <position x="-497.6" y="3662.9" z="364.7" />
      <position x="-497.6" y="3660.9" z="364.7" />
      <position x="-497.7" y="3658.9" z="364.7" />
      <position x="-497.7" y="3656.9" z="364.6" />
      <position x="-497.7" y="3654.9" z="364.6" />
```

注意事项：

- 路线太长专家可能失败
- 数据均衡性很重要

```
<scenarios>
  <scenario name="ParkingCutIn_1" type="ParkingCutIn">
    <direction value="left" />
    <trigger_point x="-497.7" y="3659.9" yaw="-90.2" z="364.7" />
  </scenario>
</scenarios>
<weathers>
  <weather cloudiness="100.0" fog_density="10.0" precipitation="50.0" precipitation_deposits="100.0" route_percentage="0" sun_altitude_angle="45.0" sun_azimuth_angle="-1.0" wetness="80.0"
  <weather cloudiness="100.0" fog_density="10.0" precipitation="50.0" precipitation_deposits="100.0" route_percentage="100" sun_altitude_angle="45.0" sun_azimuth_angle="-1.0" wetness="80.
</weathers>
```

端到端自动驾驶闭环评测 - 数据采集

- 数据采集 - 定义数据内容与格式：
  - ☐ 调用官方API，摆放传感器。参加比赛注意不要超过官方限制

DriveAdapter / leaderboard / team_code / roach_ap_agent_data_collection.py

Code Blame 899 lines (787 loc) · 40.8 KB

```python
class ROACHAgent(autonomous_agent.AutonomousAgent):

        actor_lis = self._get_3d_bbs(lidar=saved_lidar, max_distance=50)
        pos = self._get_position(tick_data)
        theta = tick_data['compass']
        speed = tick_data['speed']
        weather = tick_data['weather']
        data = {
                'x': pos[0],
                'y': pos[1],
                'theta': theta,
                'speed': speed,
                'x_command_far': far_node[0],
                'y_command_far': far_node[1],
                'command_far': far_command.value,
                'x_command_near': near_node[0],
                'y_command_near': near_node[1],
                'command_near': near_command.value,
                'should_brake': should_brake,
                'x_target': tick_data['x_target'],
                'y_target': tick_data['y_target'],'target_command': tick_data['next_command'],
                'weather': weather,
                "acceleration":tick_data["acceleration"].tolist(),
                "angular_velocity":tick_data["angular_velocity"].tolist()
                }

        if self.is_local:
            outfile = open(self.save_path / '3d_bbs' / ('%04d.json' % frame), 'w')
            json.dump(actor_lis, outfile, indent=4, default=np_encoder)
            outfile.close()

            outfile = open(self.save_path / 'measurements' / ('%04d.json' % frame), 'w')
            json.dump(data, outfile, indent=4)
```

```python
def sensors(self):
    sensors = [
            # camera rgb
            {
                'type': 'sensor.camera.rgb',
                'x': 0.80, 'y': 0.0, 'z': 1.60,
                'roll': 0.0, 'pitch': 0.0, 'yaw': 0.0,
                'width': 1600, 'height': 900, 'fov': 70,
                'id': 'CAM_FRONT'
                },
            {
                'type': 'sensor.camera.rgb',
                'x': 0.27, 'y': -0.55, 'z': 1.60,
                'roll': 0.0, 'pitch': 0.0, 'yaw': -55.0,
                'width': 1600, 'height': 900, 'fov': 70,
                'id': 'CAM_FRONT_LEFT'
                },
            {
                'type': 'sensor.camera.rgb',
                'x': 0.27, 'y': 0.55, 'z': 1.60,
                'roll': 0.0, 'pitch': 0.0, 'yaw': 55.0,
                'width': 1600, 'height': 900, 'fov': 70,
                'id': 'CAM_FRONT_RIGHT'
                },
```

```python
def tick(self, input_data):
    # control
    control = self.manager.ego_vehicles[0].get_control()

    # camera_bgr
    cam_bgr_front = input_data['CAM_FRONT'][1][:, :, :3]
    cam_bgr_front_left = input_data['CAM_FRONT_LEFT'][1][:, :, :3]
    cam_bgr_front_right = input_data['CAM_FRONT_RIGHT'][1][:, :, :3]
    cam_bgr_back = input_data['CAM_BACK'][1][:, :, :3]
    cam_bgr_back_left = input_data['CAM_BACK_LEFT'][1][:, :, :3]
    cam_bgr_back_right = input_data['CAM_BACK_RIGHT'][1][:, :, :3]
    cam_bgr_top_down = input_data['TOP_DOWN'][1][:, :, :3]

    # radar
    radar_front = input_data['RADAR_FRONT'][1].astype(np.float16)
    radar_front_left = input_data['RADAR_FRONT_LEFT'][1].astype(np.float16)
    radar_front_right = input_data['RADAR_FRONT_RIGHT'][1].astype(np.float16)
    radar_back_left = input_data['RADAR_BACK_LEFT'][1].astype(np.float16)
    radar_back_right = input_data['RADAR_BACK_RIGHT'][1].astype(np.float16)

    # lidar
    lidar = input_data['LIDAR_TOP']
    lidar_seg = input_data['LIDAR_TOP_SEG']
```

端到端自动驾驶闭环评测 – 数据采集

- 数据采集 – 定义数据内容与格式：
  - 定义所需标注：从传感器或API

```
},
{
    'type': 'sensor.camera.depth',
    'x': -0.32, 'y': 0.55, 'z': 1.60,
    'roll': 0.0, 'pitch': 0.0, 'yaw': 110.0,
    'width': 1600, 'height': 900, 'fov': 70,
    'id': 'CAM_BACK_RIGHT_DEPTH'
    },
# camera seg
{
    'type': 'sensor.camera.semantic_segmentation',
    'x': 0.80, 'y': 0.0, 'z': 1.60,
    'roll': 0.0, 'pitch': 0.0, 'yaw': 0.0,
    'width': 1600, 'height': 900, 'fov': 70,
    'id': 'CAM_FRONT_SEM_SEG'
    },
{
    'type': 'sensor.camera.semantic_segmentation',
    'x': 0.27, 'y': -0.55, 'z': 1.60,
    'roll': 0.0, 'pitch': 0.0, 'yaw': -55.0,
    'width': 1600, 'height': 900, 'fov': 70,
    'id': 'CAM_FRONT_LEFT_SEM_SEG'
    },
```

```python
def get_bounding_boxes(self, lidar=None, radar=None):
    results = []

    # ego_vehicle
    npc = self.manager.ego_vehicles[0]
    npc_id = str(npc.id)
    npc_type_id = npc.type_id
    npc_base_type = npc.attributes['base_type']
    location = npc.get_transform().location
    rotation = npc.get_transform().rotation
    #
    # verts = [v for v in npc.bounding_box.get_world_vertices(npc.get_transform())]
    # center, extent = get_center_and_extent(verts)
    # from carla official
    # bb_cords = _bounding_box_to_world(npc.bounding_box)
    # world_cord = _vehicle_to_world(bb_cords, npc)
    # from handcraft
    extent = npc.bounding_box.extent
    center = npc.get_transform().transform(npc.bounding_box.location)
    local_verts = calculate_cube_vertices(npc.bounding_box.location, npc.bounding_box.extent)
    global_verts = []
    for l_v in local_verts:
        g_v = npc.get_transform().transform(carla.Location(l_v[0], l_v[1], l_v[2]))
        global_verts.append([g_v.x, g_v.y, g_v.z])
    ###################
    ego_speed = self._get_forward_speed(transform=npc.get_transform(), velocity=npc.get_velocity())
    ego_brake = npc.get_control().brake
    ego_matrix = np.array(npc.get_transform().get_matrix())
    ego_yaw = np.deg2rad(rotation.yaw)
    road_id = CarlaDataProvider.get_map().get_waypoint(location).road_id
```

```
VehicleTurningRoute_Town15_Route480_Weather18
VehicleTurningRoute_Town15_Route504_Weather10
VehicleTurningRoute_Town15_Route519_Weather25
YieldToEmergencyVehicle_Town03_Route148_Weather18
YieldToEmergencyVehicle_Town04_Route165_Weather7
```

```
anno  camera  expert_assessment  lidar  radar  result.json
```

```
00000.jpg  00010.jpg  00020.jpg  00030.jpg  00040.jpg  00050.jpg
00001.jpg  00011.jpg  00021.jpg  00031.jpg  00041.jpg  00051.jpg
00002.jpg  00012.jpg  00022.jpg  00032.jpg  00042.jpg  00052.jpg
00003.jpg  00013.jpg  00023.jpg  00033.jpg  00043.jpg  00053.jpg
00004.jpg  00014.jpg  00024.jpg  00034.jpg  00044.jpg  00054.jpg
00005.jpg  00015.jpg  00025.jpg  00035.jpg  00045.jpg  00055.jpg
00006.jpg  00016.jpg  00026.jpg  00036.jpg  00046.jpg  00056.jpg
00007.jpg  00017.jpg  00027.jpg  00037.jpg  00047.jpg  00057.jpg
00008.jpg  00018.jpg  00028.jpg  00038.jpg  00048.jpg  00058.jpg
00009.jpg  00019.jpg  00029.jpg  00039.jpg  00049.jpg  00059.jpg
```

端到端自动驾驶闭环评测 – 数据采集

- 数据采集 – 定义数据内容与格式:
  - 设计专家模型：有了路线之后，官方并不提供具体开法。需要自行定义。不同于评测时，我们可以获得仿真中的所有信息。
    - 规则开车
    - 强化学习训练模型
    - 手动开车
- 并行启动，CARLA中的采集由于需要逐帧渲染，使用时间较长。例如：
  Bench2Drive base subset (1000 clips) == 8卡2周
  注意：数据会占用几百GB到几十TB空间，设计好压缩算法且准备好硬盘
- 后处理，CARLA会出现bug， 专家模型可能并不完美。需要剔除掉有违规的路线（错误样本影响模仿学习性能）以及渲染出错的样本

端到端自动驾驶闭环评测 – 模型训练

- 模型训练：模仿学习的训练是纯监督学习概念，可以使用任意经典深度学习框架。只需定义好Dataset类即可

Bench2DriveZoo / mmcv / datasets / B2D_e2e_dataset.py

Code Blame 855 lines (758 loc) · 37.4 KB

```python
class B2D_E2E_Dataset(Custom3DDataset):
    def get_map_info(self, index):
    def get_map_info(self, index):

        gt_masks = []
        gt_labels = []
        gt_bboxes = []

        ann_info = self.data_infos[index]
        town_name = ann_info['town_name']
        map_info = self.map_infos[town_name]
        lane_points = map_info['lane_points']
        lane_sample_points = map_info['lane_sample_points']
        lane_types = map_info['lane_types']
        trigger_volumes_points = map_info['trigger_volumes_points']
        trigger_volumes_sample_points = map_info['trigger_volumes_sample_points']
        trigger_volumes_types = map_info['trigger_volumes_types']
        world2lidar = np.array(ann_info['sensors']['LIDAR_TOP']['world2lidar'])
        ego_xy = np.linalg.inv(world2lidar)[0:2,3]

        #1st search
        max_distance = 100
        chosed_idx = []
        for idx in range(len(lane_sample_points)):
            single_sample_points = lane_sample_points[idx]
            distance = np.linalg.norm((single_sample_points[:,0:2]-ego_xy),axis=-1)
            if np.min(distance) < max_distance:
                chosed_idx.append(idx)

        for idx in chosed_idx:
            if not lane_types[idx] in self.map_element_class.keys():
                continue
```

端到端自动驾驶闭环评测 – 模型评测

- 模型评测：
  - 模型评测需要在指定的路线上完成驾驶，根据路线完成度与违规多少综合考量分数

Evaluation and metrics

The driving proficiency of an agent can be characterized by multiple metrics. For this leaderboard we have selected a set of metrics that help understand different aspects of driving. While all routes have the same type of metrics, their respective values are calculated separately. The specific metrics are as follows:

- **Driving score**: $R_i P_i$, — Main metric of the leaderboard, serving as the product between the route completion and the infractions penalty. Here Ri is the percentage of completion of the $i-th$ route, and Pi, the infraction penalty of the $i-th$ route.
- **Route completion**: Percentage of the route distance completed by an agent.
- **Infraction penalty**: $\prod_j^{\text{ped., ..., stop}} (p_i^j)^{\#\text{infractions}_j}$. — The leaderboard tracks several types of infractions and this metric aggregates all of these infractions triggered by an agent as a geometric series. Agents start with an ideal **1.0** base score, which is reduced each type an infraction is commited.

When all routes have been completed, a global metric for each of the previous three types is also generated, being the arithmetic mean of all the individual routes combined. **The global driving score is the main metric on which you will be classified with respect to other participants.**

Besides these, there is one additional infraction which has no coefficient, and instead affects the computation of the route completion (R_i).

- **Off-road driving** — If an agent drives off-road, that percentage of the route will not be considered towards the computation of the route completion score.

Additionally, some events will interrupt the simulation, preventing the agent to continue. In these cases, the route which is being simulated will be shut down, and the leaderboard will move onto the next one, triggering it normally.

- **Route deviation** — If an agent deviates more than `30` meters from the assigned route.
- **Agent blocked** — If an agent doesn't take any actions for `180` simulation seconds.
- **Simulation timeout** — If no client-server communication can be established in `60` seconds.
- **Route timeout** — If the simulation of a route takes too long to finish.

- **Collisions with pedestrians** — `0.50`.
- **Collisions with other vehicles** — `0.60`.
- **Collisions with static elements** — `0.65`.
- **Running a red light** — `0.70`.
- **Running a stop sign** — `0.80`.

Some scenarios feature behaviors that can block the ego-vehicle indefinitely. These scenarios will have a timeout of 4 minutes after which t ego-vehicle will be released to continue the route. However, a penalty is applied when the time limit is breached:

- **Scenario timeout** — `0.7`

The agent is expected to maintain a minimum speed in keeping with nearby traffic. The agent's speed will be compared with the speed of nearby vehicles. Failure to maintain a suitable speed will result in a penalty. The penalty applied is dependent on the magnitude of the spee difference, up to the following value:

- **Failure to maintain minimum speed** — `0.7`

The agent should yield to emergency vehicles coming from behind. Failure to allow the emergency vehicle to pass will incur a penalty:

- **Failure to yield to emergency vehicle** — `0.7`

Besides these, there is one additional infraction which has no coefficient, and instead affects the computation of the route completion (R_i).

- **Off-road driving** — If an agent drives off-road, that percentage of the route will not be considered towards the computation of the route completion score.

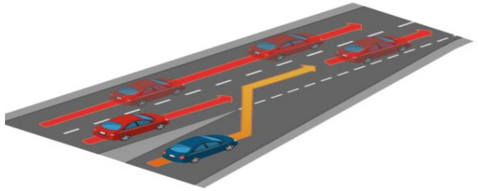
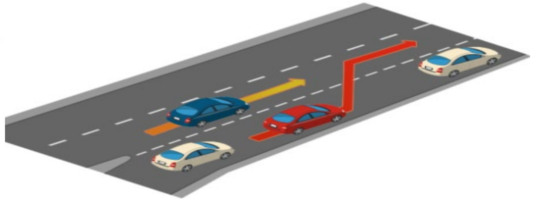
端到端自动驾驶闭环评测 – 模型评测

- 模型评测:
  - CARLA只是一个仿真器，本身并不提供任何自车/他车位置行为相关功能

```
cmd = 'DISPLAY= bash ' + os.path.join(self.carla_path, 'CarlaUE4.sh')+f' -opengl -carla-rpc-port={args.port} -nosound'
```

```
4.24.3-0+++UE4+Release-4.24 518 0
Disabling core dumps.
```

  - CARLA官方团队为模型评测另外写了两个库:
    - Leaderboard: 主进程，负责整个评测逻辑，包括读取所有路线，在CARLA中加载路线，运行他车，给自车提供传感器信息等。
    - Scenario Runner: 该库包含了从现实世界中抽象出来的种种突发事件与强交互场景，负责在CARLA中摆放与控制这些场景

端到端自动驾驶闭环评测 － 模型评测

- 模型评测：
  - 制定评测城镇、路线、天气、事件
  - 撰写Team Agent – 连接模型与CARLA （不同于专家模型，此时不能调用任何API获得传感器信息之外的数据）
    - 制定所需传感器
    - 仿照训练时的预处理，进行数据预处理
    - 数据输入模型，推理结果
    - 根据结果进行后处理（e.g.，轨迹转控制信号，紧急制动，etc）
  - （可选）多卡并行评测
  - 计算总分

端到端自动驾驶闭环评测 - FAQ

- FAQ：
  - 我需要自己编译CARLA嘛？

    如果仅是训练与测试模型，不需要，下载官方编译好的版本即可。如果需要自定义素材/场景，需要几百G空间以及熟悉c++编译相关。
  - CARLA crash了怎么办？

    CARLA crash是常见现象。需自行写好鲁棒代码，支持断点续传。此外，CARLA对系统中的一些库以及python库的版本有要求，具体参考官方，以及自行逐行实验。由于是C++底层，没有任何报错信息是正常现象
  - 有什么推荐的CARLA中的闭环模型嘛？

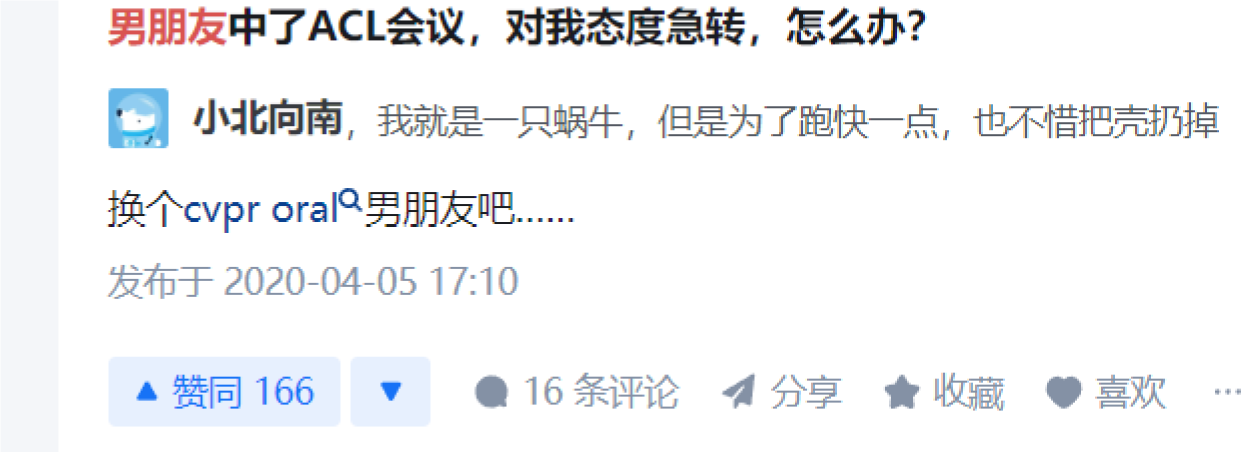
    TCP (NeurIPS 22)：简单易上手，3090即可训练。

    ThinkTwice (CVPR 23)/DriveAdapter(ICCV 23 Oral)：冲击SOTA，需要大硬盘 + 多机A100

    Bench2Drive：提供UniAD和VAD的CARLA实现

端到端自动驾驶闭环评测

- 我有一个绝妙的idea（可惜这里空白太小写不下），需要3个月内发一篇CVPR/NeurIPS?



- Reviewer要求我补闭环实验，我只有一周时间？而且我并没有baseline的闭环点数？
- 我的模型怎么也跑不过baseline，自采数据的分布或者专家有问题？
- CARLA官方route怎么方差这么大？Leaderboard v2上大家怎么只有个位数的分数？
- 老板要求把UniAD三天内跑起来看看效果？

端到端自动驾驶闭环评测开源框架 – Bench2Drive

现有的开源端到端评测框架：

- 开环评测（log replay）难以体现真实驾驶技能

 -> **设计CARLA仿真器中的闭环评测框架**

- 闭环评测路线难度相对较低

 -> **整理44种真实世界corner case进行评测**

- 闭环评测缺乏统一的数据集，各团队各自采集

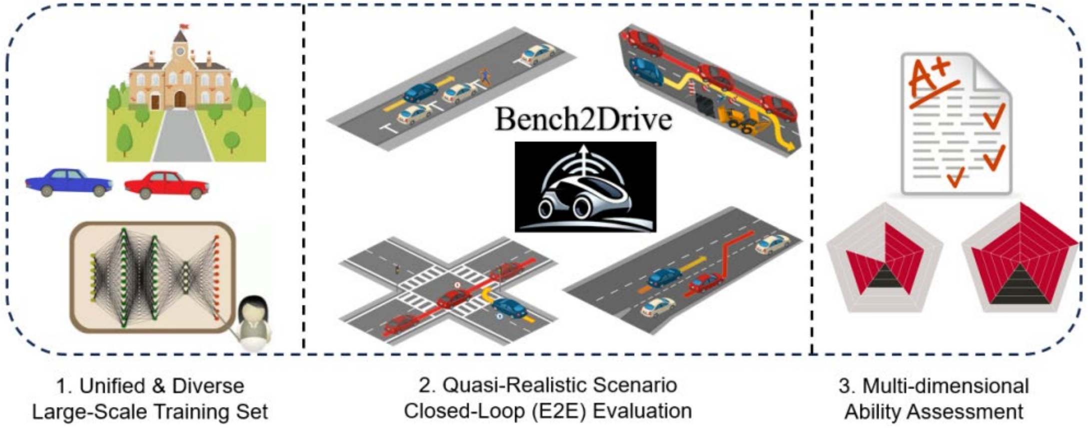
 -> **大规模、多样化、全面标注的10K段数据，每段包含150-200米的里程，覆盖44种交互场景，23种天气，12座城市，标注包含2D/3D框、语义分割、深度估计以及专家模型评估**

- 已有框架单纯把全部路线评分取平均，难以分析不同驾驶系统的具体能力的特点

 -> **按能力分门别类细致打分**

Table 1: **Comparison with related planning benchmarks.**

Benchmark	Sensor	Closed-Loop	E2E-Sim	Expert	Complex	Multi-Ability-Eval
nuScenes [2]	✓	✗	✗	✓	✗	✗
nuPlan [28]	✓	✓	✗	✓	✓	✗
Waymax [14]	✗	✓	✗	✓	✓	✗
Longest6 [7]	✓	✓	✓	✓	✗	✗
CARLA LB V2 [12]	✓	✓	✓	✗	✓	✗
Bench2Drive (Ours)	✓	✓	✓	✓	✓	✓



Bench2Drive: Towards Multi-Ability Benchmarking of Closed-Loop End-To-End Autonomous Driving, submitted to NeurIPS 2024 dataset and benchmark track

端到端自动驾驶闭环评测开源框架 - Bench2Drive

开源端到端自动驾驶框架 https://github.com/Thinklab-SJTU/Bench2DriveZoo

- 将mm系列（mmcv/mmdet/mmdet3d）等自动驾驶常用库进行**剥离**，降低与深度学习框架的耦合程度，不强行限制Pytorch版本（FA2/ThunderKittens in UniAD）
- 提供6个前沿端到端自动驾驶方法的指标，包括UniAD/VAD在CARLA训练+评测代码

Table 4: **Multi-Ability Results of E2E-AD Methods.** * denotes expert feature distillation.

Method	Ability (%) ↑					
	Merging	Overtaking	Emergency Brake	Give Way	Traffic Sign	**Mean**
AD-MLP [51]	0.00	0.00	0.00	0.00	0.00	0.00
UniAD-Tiny [17]	4.11	12.50	14.54	10.00	18.54	11.94
UniAD-Base [17]	9.46	12.50	20.00	30.00	23.03	19.00
VAD [27]	0.13	17.50	14.54	30.00	25.55	20.02
TCP* [47]	8.70	10.00	7.27	10.00	7.95	8.78
TCP-ctrl*	8.57	10.00	3.63	0.00	7.95	6.03
TCP-traj*	24.29	15.00	29.09	**50.00**	51.67	34.01
ThinkTwice* [25]	26.44	17.50	32.12	**50.00**	53.65	35.94
DriveAdapter* [23]	**29.23**	**20.00**	**34.71**	**50.00**	**57.21**	**38.23**

Method	mAP	NDS	Config	Download
BEVFormer-Tiny	0.37	0.43	config	Hugging Face/Baidu Cloud
BEVFormer-Base	0.63	0.67	config	Hugging Face/Baidu Cloud

端到端自动驾驶闭环评测开源框架 - Bench2Drive

✓ 安装CARLA

Setup

- Download and setup CARLA 0.9.15

```
mkdir carla
cd carla
wget https://carla-releases.s3.us-east-005.backblazeb2.com/Linux/CARLA_0.9.15.tar.gz
tar -xvf CARLA_0.9.15.tar.gz
cd Import && wget https://carla-releases.s3.us-east-005.backblazeb2.com/Linux/AdditionalMaps_0.
cd .. && bash ImportAssets.sh
export CARLA_ROOT=YOUR_CARLA_PATH
echo "$CARLA_ROOT/PythonAPI/carla/dist/carla-0.9.15-py3.7-linux-x86_64.egg" >> YOUR_CONDA_PATH/
```

✓ 下载数据集

Dataset

- The datasets has 3 subsets, namely Mini (10 clips), Base (1000 clips) and Full (10000 clips), to accommodate different levels of computational resource.
- Detailed explanation of dataset structure, annotation information, and visualization of data.

Subset	Hugging Face🤗	Baidu Cloud	Approx. Size
Mini	Download script	Download script	4G
Base	Hugging Face Link	Baidu Cloud Link	400G
Full	Hugging Face Link	Uploading	4T

端到端自动驾驶闭环评测开源框架 - Bench2Drive

✓ 配置目录

Download Bench2Drive

Download our dataset from (LINK) and make sure the structure of data as follows:

```
Bench2DriveZoo
├── ...
├── data/
|   ├── bench2drive/
|   |   ├── v1/                                          # Bench2Drive base
|   |   |   ├── Accident_Town03_Route101_Weather23/
|   |   |   ├── Accident_Town03_Route102_Weather20/
|   |   |   └── ...
|   |   └── maps/                                        # maps of Towns
|   |       ├── Town01_HD_map.npz
|   |       ├── Town02_HD_map.npz
|   |       └── ...
|   ├── others
|   |       └── b2d_motion_anchor_infos_mode6.pkl        # motion anchors for UniAD
|   └── splits
|           └── bench2drive_base_train_val_split.json    # trainval_split of Bench2Drive base
```

✓ 生成nuScenes-style 数据集pickle

Prepare Bench2Drive data info

Run the following command:

```
cd mmcv/datasets
python prepare_B2D.py --workers 16   # workers used to prepare data
```

The command will generate `b2d_infos_train.pkl`, `b2d_infos_val.pkl`, `b2d_map_infos.pkl` under `data/infos`. *Note: It will take about 1 hour to generate all the data with 16 workers*

端到端自动驾驶闭环评测开源框架 - Bench2Drive

✓ 安装UniAD环境

Bench2DriveZoo / docs / INSTALL.md

zhiyuanzzz Polish documents 3765df5 · 3

Preview Code Blame 55 lines (47 loc) · 2.15 KB Raw

Follow these steps to install the environment

- STEP 1: Create enviroment

```
conda create -n b2d_zoo python=3.8
conda activate b2d_zoo
```

- STEP 2: Install cudatoolkit

```
conda install -c "nvidia/label/cuda-11.8.0" cuda-toolkit
```

- STEP 3: Install torch

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

- STEP 4: Set environment variables

```
export PATH=YOUR_GCC_PATH/bin:$PATH
export CUDA_HOME=YOUR_CUDA_PATH/
```

- STEP 5: Install ninja and packaging

✓ 训练BEVFormer

BEVFormer

Train

```
#train BEVFormer base
./adzoo/bevformer/dist_train.sh ./adzoo/bevformer/configs/bevformer/bevformer_base_b2d.py 4 #N_GPUS
#train BEVFormer tiny
./adzoo/bevformer/dist_train.sh ./adzoo/bevformer/configs/bevformer/bevformer_tiny_b2d.py 4 #N_GPUS
```

Open loop eval

```
#eval BEVFormer base
./adzoo/bevformer/dist_test.sh ./adzoo/bevformer/configs/bevformer/bevformer_base_b2d.py ./ckpts/bevformer_base_b2d.pth 1
#test BEVFormerr tiny
./adzoo/bevformer/dist_test.sh ./adzoo/bevformer/configs/bevformer/bevformer_tiny_b2d.py ./ckpts/bevformer_tiny_b2d.pth 1
```

端到端自动驾驶闭环评测开源框架 – Bench2Drive

✓ 训练UniAD

UniAD

Train stage1

```
#train UniAD base
./adzoo/uniad/uniad_dist_train.sh  ./adzoo/uniad/configs/stage1_track_map/base_track_map_b2d.py 4
#train UniAD tiny
./adzoo/uniad/uniad_dist_train.sh  ./adzoo/uniad/configs/stage1_track_map/tiny_track_map_b2d.py 4
```

Train stage2

```
#train UniAD base
./adzoo/uniad/uniad_dist_train.sh  ./adzoo/uniad/configs/stage2_e2e/base_e2e_b2d.py 1
#train UniAD tiny
./adzoo/uniad/uniad_dist_train.sh  ./adzoo/uniad/configs/stage2_e2e/tiny_e2e_b2d.py 1
```

Open loop eval

```
#eval UniAD base
./adzoo/uniad/uniad_dist_eval.sh ./adzoo/uniad/configs/stage2_e2e/base_e2e_b2d.py ./ckpts/uniad_base_b2d.pth 1
#eval UniAD tiny
./adzoo/uniad/uniad_dist_eval.sh ./adzoo/uniad/configs/stage2_e2e/tiny_e2e_b2d.py ./ckpts/uniad_tiny_b2d.pth 1
```

端到端自动驾驶闭环评测开源框架 － Bench2Drive

✓ 链接UniAD在CARLA中的agent

Link this repo to Bench2Drive

```
# Add your agent code
cd Bench2Drive/leaderboard
mkdir team_code
ln -s Bench2DriveZoo/team_code/* ./team_code    # link UniAD,VAD agents and utils
cd ..
ln -s Bench2DriveZoo  ./                        # link entire repo to Bench2Drive.
```

Bench2DriveZoo / team_code /

zhiyuanzzz Update uniad_b2d_agent.py and vad_b2d_agent.py

Name	Last commit message
..	
pid_controller.py	add uniad/vad codebase
planner.py	add uniad/vad codebase
uniad_b2d_agent.py	Update uniad_b2d_agent.py and vad_b2d_agent.py
vad_b2d_agent.py	Update uniad_b2d_agent.py and vad_b2d_agent.py

端到端自动驾驶闭环评测开源框架 – Bench2Drive

✓ 多卡评测

- Multi-Process Multi-GPU Parallel Eval. If your team_agent saves any image for debugging, it might take lots of disk space.

```
# Please set TASK_NUM, GPU_RANK_LIST, TASK_LIST, TEAM_AGENT, TEAM_CONFIG, recommend GPU:Task(1:
bash leaderboard/scripts/run_evaluation_multi.sh
```

✓ （可选）可视化中间结果

- Visualization - make a video for debugging with canbus info printed on the sequential images.

```
python tools/generate_video.py -f your_rgb_folder/
```

✓ 计算总分

- Metric

```
# Merge eval json and get driving score and success rate
python tools/merge_reoute_json.py -f your_json_folder/

# Get multi-ability results
python tools/ability_benchmark.py -r merge.json
```

谢谢大家，欢迎讨论！