Driving SMARTS Competition at NeurIPS 2022: Insights and Outcome

smarts-project.github.io

Amir Rasouli[*] amir.rasouli@huawei.com
Soheil Alizadeh[*] soheil.shabestary@huawei.com
Iuliia Kotseruba[†] yulia_k@eecs.yorku.ca
Yi Ma[‡] mayi@tju.edu.cn
Hebin Liang[‡] lianghebin@tju.edu.cn
Yuan Tian[§] yuan.t426@gmail.com
Zhiyu Huang[¶] zhiyu001@e.ntu.edu.sg
Haochen Liu[¶] haochen002@e.ntu.edu.sg
Jingda Wu[¶] jingda001@e.ntu.edu.sg
Randy Goebel[‖] rgoebel@ualberta.ca
Tianpei Yang[‖] tianpei1@ualberta.ca
Matthew E. Taylor[‖] matthew.e.taylor@ualberta.ca
Liam Paull[**] paulll@iro.umontreal.ca
Xi Chen[*] xi.chen4@huawei.com

Editors: Marco Ciccone, Gustavo Stolovitzky, Jacob Albrecht

Abstract

The Driving SMARTS (Scalable Multi-Agent Reinforcement Learning Training School) competition was designed to address one of the major challenges for autonomous driving (AD), namely adaptation to distribution shift between data used for training and inference and the problems caused by this shift in real-world conditions.

The two key features of the competition are 1) a two-track structure to encourage and support a variety of approaches to solving the problem, such as reinforcement learning, offline learning, and other machine learning methods; and 2) curated data for driving scenarios of varying difficulty levels, from cruising to unprotected turns at unsignalized intersections.

The competition attracted 87 participants in 53 teams. Top-ranking teams contributed a diverse set of solutions highlighting the effectiveness of different methodologies on safe motion planning for AD. This paper provides an overview of the Driving SMARTS competition, discusses its organisational and design aspects, and presents the results, insights, and promising directions for future research.

Keywords: Autonomous Driving; Motion Planning; Offline Learning; Reinforcement Learning; Distribution Shift

[*] Huawei Technologies Canada
[†] York University
[‡] Deep RL Lab, Tianjin University
[§] University of British Columbia
[¶] AutoMan lab, Nanyang Technological University
[‖] University of Alberta & Alberta Machine Intelligence Institute (Amii)
[**] University of Montreal

1. Introduction

Distribution shift between data used for training and inference is one of the key challenges for machine learning in the domain of autonomous driving (AD). Whether the chosen method is reinforcement learning (RL), supervised learning, or classical planning and control, such a challenge must be properly resolved to allow the deployment of autonomous vehicles in the real-world.

The Driving SMARTS competition was designed to address the problems in AD motion planning caused by distribution shift. Competition scenarios and evaluation schemes were designed to highlight a wide range of situations that may be encountered by ADs. The goal was to engage a diverse expertise in the machine learning (ML) community to contribute a wide range of solutions, such as single- and multi-agent RL, supervised learning, and control and optimisation methods with data-driven enhancements.

The competition consisted of two tracks. The first track was **freestyle**, meaning that it was open to any method to encourage the participants to tackle real-world AD challenges regardless of their domain of expertise. The second track was designed exclusively for **offline** learning methods in order to highlight offline-to-online distribution shift and ways of dealing with it for real-world deployment.

Another goal of Driving SMARTS was to make the challenges of real-world autonomous driving R&D more accessible for ML researchers by simplifying the complex practicalities of data preprocessing, scenario construction, training and test corpora curation, evaluation, and real-world system integration. To this end, the competition provided a rich set of representative scenarios integrated within a mature simulation platform. These scenarios included multilane cruising with lane changes, on-ramp merge into a flow of background traffic with subsequent lane changes in short sequence, and turns without traffic light protection. Furthermore, naturalistic driving data were used to assess the performance, realism, and real-world relevance of the proposed solutions. As our contribution to the ML community, all data, scenarios, baselines, and winning models are made available via our open-source simulation platform SMARTS[1] Zhou et al. (2020).

This paper provides an overview of the Driving SMARTS competition design and outcomes. We begin by describing different aspects of the competition including scenario design, environment and data preparation, and evaluation metrics. This is followed by a brief introduction of the three winning competition submissions and analysis of their performance on the given driving tasks. The concluding section summarizes the results of the competition and suggests future directions.

2. The Competition

In this section, we briefly review the competition, with a particular focus on the design of the tasks and evaluation procedures. More detailed information on the competition can be found in Rasouli et al. (2022).

1. https://github.com/huawei-noah/SMARTS/

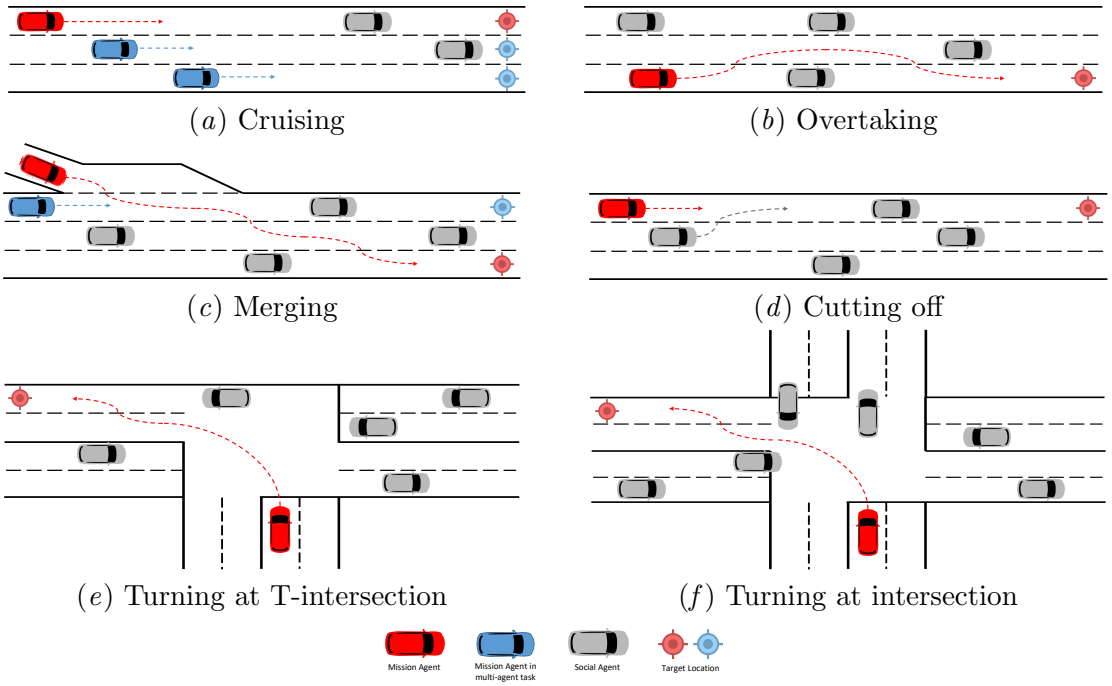


Figure 1: Illustration of the tasks used in the competition. Mission vehicle(s) that are controlled by the proposed algorithms are shown in red and blue for single- and multi-agent versions, respectively. Target symbols show locations of destinations. Some tasks, such as cruising (a) and merging (c), have both single- and multi-agent versions.

2.1. Tracks

The competition was organized into two tracks. Track 1 was freestyle and allowed participants to contribute solutions that used any methods and training data. Here, the goal was to encourage a diversity of solutions without imposing any restrictions. Track 2, on the other hand, was open exclusively to offline methods as they are a common choice for many models that rely on naturalistic data for training.

In Track 1, participants provided their inference code to be evaluated using an automated system. However, in Track 2, to ensure compliance with the offline training requirement, both the training and the inference code were submitted by the participants. The models were then trained and evaluated by the organisers.

2.2. Tasks

We designed the tasks to create a variety of degrees of difficulty for the automated agents. First, there were different types of scenario, from simple cruising, where the agent had to maintain its current path, to more challenging turning at an unsignalized intersection, which required understanding the behaviour of other agents and coordinating with them.

Within some scenarios, there were gradations of difficulty depending on the number of mission agents, from single- to multi-agent cases with up to three agents to be controlled concurrently to accomplish the task. The scenario descriptions are listed below (ordered from simpler to more challenging) and illustrated in Figure 1. The tasks are single agent by default.

- *Cruising* (single- and multi-agent) - an entry-level task to assess the stability and human-like capabilities of the method. In this task, the vehicle has to safely cruise to a predefined location without the need for any specific maneuvers. In the multi-agent setting, all agents have to arrive to their respective destinations.
- *Overtaking* - tests the capability of the agent to maneuver around slow traffic to reduce its travel time.
- *Merging* (single- and multi-agent) - the agent has limited time to perform a lane changing maneuver to merge into traffic travelling in another lane, otherwise an accident may occur. In the multi-agent task, merging vehicles as well as the vehicles travelling in the neighboring lane should be controlled.
- *Cutting off* - the agent is cut off aggressively by another vehicle controlled by the simulator and has to take proper action to avoid an accident (*e.g.*, by braking or changing lane).
- *Left turn at unsignalized T-intersection* - tests the ability of the method to follow the stop sign, estimate time to cross the intersection, turn into the correct lane, and accelerate, all without causing accidents.
- *Left/right turn at unsignalized intersection* - this task requires the agent to be aware of the traffic travelling in the opposite direction and encourages riskier maneuvers that might trigger reactions of background vehicles.

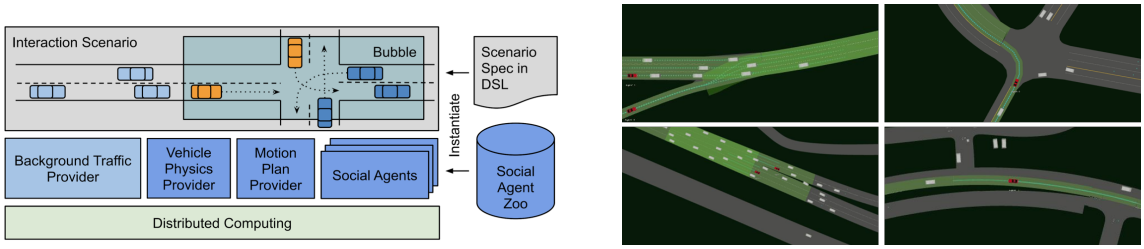
2.3. Environment and Data

2.3.1. Simulation Platform: SMARTS

For the purpose of the competition we relied on the Scalable Multi-Agent Reinforcement Learning Training School (SMARTS) Zhou et al. (2020) simulation platform, designed for research on AD systems. SMARTS was used both for generating training data and as an environment to evaluate the submitted solutions. The key characteristics of SMARTS that make it suitable for this competition and research on AD in general include *compositional architecture* that allows creation of scenarios from individual elements such as maps, traffic, *etc.*; *distributed computing* for scalable simulation; and *realistic interactions* with accurate physics and involving rule-based and data-driven social agents. A schematic diagram of the SMARTS architecture is shown in Figure 2(*a*).

2.3.2. Synthetic Data

Synthetic data are composed of hand-crafted maps that are populated with SMARTS social agents acting as background traffic. The social agents' behaviours are controlled using a built-in tool in SMARTS that generates realistic actions/reactions according to the current



(*a*) SMARTS architecture (*b*) Scenarios in SMARTS

Figure 2: a) A schematic diagram of the SMARTS platform architecture. b) Examples of generated synthetic (top row) and real (bottom row) scenarios.

status of the traffic. This provides a means of modelling interactions between mission agents and surrounding traffic. Some examples of synthetic scenarios are shown in Figure 2(*b*) (top row).

2.3.3. Naturalistic Data

SMARTS also provides a tool for generating realistic scenarios using replay of data from naturalistic datasets. At the time of competition, two datasets were supported, Waymo Open Dataset Sun et al. (2020) and NGSIM Kovvali et al. (2007), both of which were used for evaluation. To create naturalistic scenarios, samples were selected from the datasets and replayed in the SMARTS environment. Actions of surrounding traffic agents were imported from the historical data, therefore the mission agent had to complete its task without interaction with the traffic. Examples of naturalistic scenarios are shown in Figure 2(*b*) (bottom row).

2.3.4. Scenarios

Overall, we generated 128 different scenarios that cover all types of tasks defined in Section 2.2 with different levels of difficulty depending on traffic volume, road structure, and destination points. There were 57 non-reactive scenarios derived from natural datasets. The rest were synthetic with reactive background traffic consisting of SMARTS-controlled agents.

2.4. Metrics

For evaluation, we used four metrics designed to highlight different aspects of methods' performance. The metrics are listed below in the order of importance. The submissions were ranked based on this order and in case of a tie, the next metric in the hierarchy was used to break the tie. Smaller values are better for all metrics.

1. **Completion** captures how many scenarios have been successfully completed. It is converted into an error measure as follows:

$$\texttt{completion} = \frac{\texttt{num. failures}}{\texttt{num. sc}}$$

where *sc* is `scenario`.

2. **Time** is the average time to finish a scenario. This metric captures the efficiency of the generated path as follows:

$$\texttt{time} = \frac{\sum_{sc} T^i}{\texttt{num. sc}}$$

where T^i is the number of time steps to complete scenario i.

3. **Humanness** measures driving similarity to that of human drivers. While frequent direction changes, hard acceleration, and braking might score higher on other metrics, they are not human-like. The humanness metric is computed as:

$$\texttt{humanness} = \frac{\sum_{T} \texttt{distance}_{obs} + \texttt{jerk} + lc_{off}}{\sum_{sc} T^i}$$

where *obs* is obstacle, jerk is the rate of change in vehicle's acceleration between consecutive time points, and lc_{off} is lane center offset given by,

$$lc_{off} = \left(\frac{\texttt{distance from center}}{\texttt{lane width}}\right)^2.$$

4. **Rule violations** assess whether the agents follow traffic laws. We consider two violations: exceeding speed limits and driving in the wrong direction:

$$\texttt{rule} = \frac{\sum_{sc}\sum_{T^i} min\left(\frac{s_{violate}}{0.5 \cdot s_{limit}}, 1\right) + \texttt{road}_{violate}}{\sum_{sc} T^i}$$

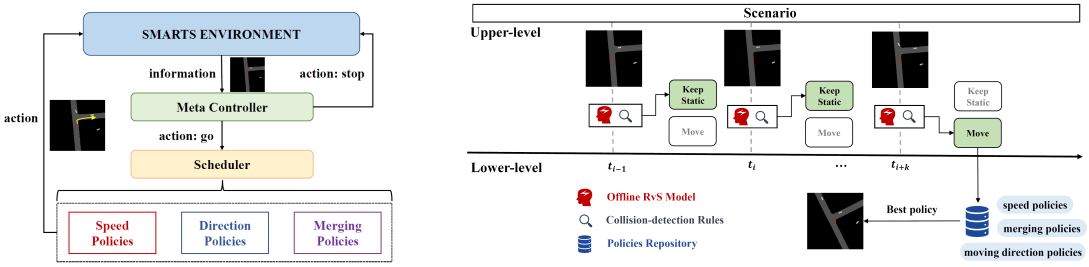
where $s_{violate} = max(0, s_a - s_{limit})$ measures speeding (*i.e.*, speed over the posted limit) and $\texttt{road}_{violate} \in \{0, 1\}$ is whether the road direction was violated.

Vehicle collisions are detected when the bounding box of the mission agent overlaps with that of another vehicle (*e.g.,* another mission agent or traffic vehicle). Mission agent involved in collisions is removed in subsequent time steps. Its partial trajectory still contributes to humanness and rule violations metrics, computed as averages over timesteps and number of agents.

3. Competition Submissions

A total of 87 individuals in 53 teams participated in the competition and contributed 18 valid submissions. The top three teams, **tjudrllab-fanta**, **VCR**, and **AID**, proposed offline learning methods and were eligible to compete in both tracks. Before discussing the winning teams' submissions[2], we briefly describe the baseline method that was provided as a sample solution.

2. Detailed descriptions of the methods are available on the competition webpage at https://smarts-project.github.io/



(*a*) Architecture overview (*b*) Decision making steps

Figure 3: An overview of the model proposed by team tjudrllab-fanta.

3.1. Baseline

The baseline for the competition is an online RL model based on Proximal Policy Optimization (PPO) Schulman et al. (2017) which is an actor-critic method capable of handling continuous and discrete action spaces. The input to the model is comprised of a bird's-eye view (BEV) image of the traffic scene with superimposed waypoints, distance to the agent's goal, and the angle between the current heading of the ego vehicle and the goal position.

The mission agent's action space is discretized into four predefined actions: brake, move forward, turn right, and turn left. The agent receives a reward signal that combines penalties for going off road, driving on the wrong side, and collisions, a large reward for reaching the goal location, and another reward for the distance travelled.

3.2. 1st Place: tjudrllab-fanta (TF)

The model proposed by team TF follows a hierarchical structure as illustrated in Figure 3(*a*). Given the information from the SMARTS simulator, an upper-level module (the *Meta Controller*) decides whether to pass the information to the lower-level module (the *Scheduler*) to make further decisions. If the Scheduler is activated, it produces specific driving decisions and returns them to SMARTS. Otherwise, an *Emergency Break* decision is made.

The Meta Controller is in charge of deciding whether the mission agent should move or remain static. This module consists of an **offline learning agent**, trained using reinforcement learning via supervised learning (RvS) Emmons et al. (2021), and a rule-based collision detection module. The Scheduler maintains a policy repository with three policies that control the agent's speed, heading, and lane changes (see Figure 3(*b*)).[3]

Meta Controller. The Meta Controller uses a rule-based collision detection module to detect vehicles that might cause incidents with the agent. This module is augmented with an additional collision detector using an offline agent trained using RvS method.

The RvS method follows a Markov decision process (MDP) design. *States* are represented with information about the agent and its surroundings (e.g., an agent's bounding

3. Additional information and implementation is available at https://github.com/superCat-star/fanta-code

box and speed), and the dynamics of the closest vehicles. *Actions* are either to move or remain stationary. The *goals* are defined as waypoints and headings in the next 5 time steps.

Training data is processed as follows: abnormal headings are corrected, data corresponding to core tasks is selected, waypoints are simulated if missing, speed information is discretized, and irrelevant vehicles (*e.g.* parked ones) are removed.

Scheduler. The Scheduler module executes the decision policy provided by the Meta Controller using the following policies:

- The *speed policy* computes the speed as a product of a baseline speed (based on the curvature of the lane where the agent is currently located) and speed attenuation coefficient (based on traffic density near the agent).
- The *moving direction policy* maintains the agent in its current lane, unless a lane change command is issued, in which case the agent sets the target lane to the adjacent one in the appropriate direction.
- The *merging policy* estimates the expected speed for the current and adjacent lanes using the speeds of vehicles within the certain range. To mitigate collision risk and avoid speed loss due to lane changes, a penalty term is imposed for the expected speed of two adjacent lanes.

3.3. 2nd Place: VCR

The model proposed by team VCR consists of two controllers: *base* and *filtering*. The base controller generates future waypoints given the observations. A Bezier curve is fit to the waypoints to generate a smooth path, from which the speed and the next goal position of the agent vehicle are selected. The filter controller receives a local BEV image of the scene together with the agent's information. Based on this input, it determines the safety of the proposed speed and classifies the action of the vehicle into one of the 6 categories: collision, wrong way, on shoulder, off road, off route, and safe.[4]

Baseline Controller. Baseline controller relies on the waypoints provided in the observation space. Since not all waypoints lead to the final goal position, the following policy is implemented to select the correct set of waypoints. First, the lane index of the final destination is determined and then the distance between the destination location and the last waypoint of each path is calculated. Based on this information, lane index is selected for the mission agent. To choose the next waypoint on the given lane, additional constraints are imposed to prevent irregular behaviors, such as driving perpendicular to the lane caused by selecting a waypoint too close to the vehicle or exceeding speed limit caused by waypoints too far away.

Filtering Controller. Given the baseline policy action, the filtering controller selects n samples from the line between the current mission vehicle's position and the next. Then, a pretrained neural network module is used to score the probability of collision or other termination events caused by executing those actions in the next time step. Based on these samples, the actions might be re-sampled again. This process is demonstrated in Figure 4.

4. The implementation of the method is available at https://github.com/yuant95/SMARTS_VCR

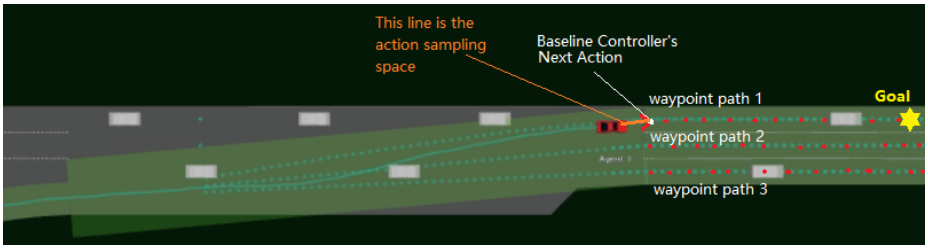
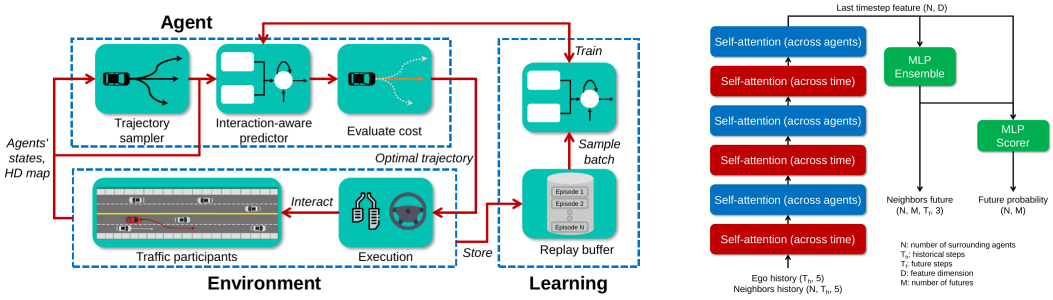


Figure 4: Illustration of sampling in filtering controller proposed by team VCR.



(*a*) Overall architecture (*b*) Predictor module

Figure 5: An overview of the model by team AID.

Action classifier This module categorizes the actions into one of the six classes mentioned earlier using a multi-class classifier composed of a convolutional neural network (CNN) and a multi-layer perceptron (MLP). The input to the CNN consists of a BEV image of the scene centered at the mission vehicle and the mission vehicle's current position. The next 5 waypoints and the vehicle's action (position and heading angle) are fed to the corresponding networks. The outputs of the networks are combined and fed into another MLP for the final classification.

3.4. 3rd Place: AID

The solution proposed by team AID is a hybrid model consisting of a Transformer-based *motion predictor* and a *sampling-based planner* (see Figure 5). The former is responsible for forecasting the future trajectories of social agents surrounding the mission vehicle, and the latter is in charge of selecting the optimal trajectory considering the distance to the goal, ride comfort, and safety. The prediction/planning horizon is set at 3 seconds, but only the first five steps (0.5s) of the planned trajectory are executed.[5]

Motion Predictor. The motion predictor receives the information for the mission vehicle and five surrounding vehicles as input and predicts the future trajectories of the neighbouring vehicles. The state of an agent at a given time step consists of its x and y coordinates, heading angle, and velocities along the x and y axes. The network employs factorised attention along the time and agent axes to exploit dependencies between the agents (see Figure

5. The implementation is available at https://github.com/MCZhi/Predictive-Decision

5(*b*)). The last time step of the feature of each surrounding agent is selected to create an ensemble of MLP heads to predict multi-modal trajectories and an MLP-based scorer to predict the probability of each trajectory.

Planner. The planner first generates a set of candidate trajectories based on the the mission vehicle's current state, available routes, and target speed profiles. The trajectories are first decoupled into longitudinal and lateral directions in the Frenét frame Werling et al. (2010) to generate different speed profiles and trajectories. These are then combined and transformed back to Cartesian space. Next, the planner obtains the prediction results from the predictor and selects the most likely trajectory for each agent. To check whether the trajectory leads to a collision, a cost function is used that takes into account distance to the goal, longitudinal jerk, lateral acceleration, collisions, distance to other agents, and time-to-collision. At the end, the planner outputs the optimal trajectory to the controller. If all the candidate trajectories cause collisions with other agents, an emergency braking command is outputted instead.

Training Process. An agent equipped with the predictor and planner manages the interaction with the environment. An episodic buffer is used to store the trajectories of all agents in the environment and load data into a replay buffer at the end of each episode. At each training step, a batch of data is sampled from the replay buffer from different episodes and time steps. The historical trajectories of the agents and their ground truth future trajectories at each time step are obtained from the episodic memory. A vehicle is randomly selected as the ego vehicle and its surrounding vehicles are determined to improve the generalisation ability of the predictor. The position and heading attributes of the input data are normalised according to the state of the ego vehicle at the given time step. The training loss is the sum of trajectory regression loss (smooth L1 loss) on the closest-to-ground-truth trajectory and cross-entropy loss of the predicted scores. The closest-to-ground truth mode is the target.

3.5. Discussion

All three winning methods discussed in the previous section were offline, however, each had a unique design using a variety of techniques, from classical offline RL methods to hybrid predictive-planning approaches. Since task completion was the primary criterion for determining the winner, the methods were optimised to complete the tasks while sacrificing performance on other metrics such as time and driving style. This was particularly apparent in the performance of the top two solutions proposed by teams TF and VCR teams, which often generated illegal behaviours and unnatural driving actions, such as sudden changes in the heading angle, acceleration / deceleration, *etc.*. The AID model, on the other hand, achieved comparable results on the task completion metric, while also performing very well on other metrics. Specifically, it had similar or better completion time to the other models and significantly better performance on the humanness metric (up to 92%) and rule violation (up to 98%).

All things considered, the solution proposed by the AID team was more successful. Its balanced performance can be partly attributed to the use of the planner module which enforces conformity of the generated trajectories to naturalistic behaviors via a carefully

designed cost function. Performance of the other two models demonstrated that purely learning-based approaches are less efficient in extracting implicit information from the training data, especially when the number of constraining criteria is large.

In terms of the completion of different driving tasks, there were no specific patterns in the performance of the methods, *i.e.*, no link could be established between the nature of the tasks and the success rate of a given method. There were, however, a number of exceptions. As anticipated, the nonreactive naturalistic scenarios were the most challenging, and all methods failed to complete approximately half of the given scenarios. Rear-end accidents were the most common problems, as all methods were designed to behave conservatively. In reactive scenarios, most errors occurred in multi-agent merging tasks where, in addition to safe planning, the agents had to cooperate with other vehicles to accomplish the task.

4. Conclusions and Future Work

The driving SMARTS competition was designed to address the distribution shift problem in the context of autonomous driving systems. We designed two tracks for the competition: Track 1 (*freestyle*) allowed participants to apply any methods to solving real-world AD challenges, and Track 2 (*offline*) tackled the question of the effectiveness of training on naturalistic AD datasets. Four metrics were proposed for evaluating the performance of the submissions, with a focus on task completion with other metrics serving as tie breakers.

A total of 53 international teams participated in the competition, of which three teams won the top spots on both tracks. All three winning solutions had unique designs that demonstrated different strengths and weaknesses on different scenarios and metrics. As such, they highlight the need for diverse ML-based solutions for motion planning and advantages and disadvantages of various techniques that will be useful for guiding future research.

Besides providing a testbed to engage researchers in tackling real-world AD challenges, Driving SMARTS made a number of contributions to the ML community, including open-source implementation of motion planning methods, a benchmark standard for evaluation of AD methods, a dataset of diverse driving scenarios, and tools for experimenting with different approaches using both naturalistic and synthetic data.

Future work Evaluation on a single metric, although it is a common approach in AD benchmarking, is not sufficient to assess the performance of the model. In addition to completing the task, the optimal algorithm should also generate behaviours that are reasonable and comply with rules and safety. Therefore, in the future, we intend to use a weighted combination of metrics that address different aspects of performance. Additional metrics can also be introduced to address other aspects of planning, such as cooperation. For example, even though the intention of the merging tasks in the competition was to encourage methods to cooperate explicitly, in practise the proposed solutions were planning individually and treating other agents as part of the regular traffic.

To engage a wider range of participants, such as novice machine learning practitioners, we intend to add simpler driving scenarios, provide a lighter and easier to work with version of the simulation platform, design tracks based on different level of task difficulty, and add additional tools and training materials for using the platform.

References

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. RvS: What is essential for offline RL via supervised learning? *arXiv:2112.10751*, 2021.

Vijay Gopal Kovvali, Vassili Alexiadis, and Lin Zhang. Video-based vehicle trajectory data collection. In *Transportation Research Board Annual Meeting*, 2007.

Amir Rasouli, Randy Goebel, Matthew E Taylor, Iuliia Kotseruba, Soheil Alizadeh, Tianpei Yang, Montgomery Alban, Florian Shkurti, Yuzheng Zhuang, Adam Scibior, et al. Neurips 2022 competition: Driving smarts. *arXiv:2211.07545*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *CVPR*, 2020.

Moritz Werling, Julius Ziegler, Soren Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *ICRA*, 2010.

Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadmellat, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedershad Banijamali, Alexander Cowen Rivers, Zheng Tian, Daniel Palenicek, Haitham bou Ammar, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving. In *CoRL*, 2020.