2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving

Napat Karnchanachari* Dimitris Geromichalos* Kok Seang Tan Nanxiang Li Christopher Eriksen
Shakiba Yaghoubi Noushin Mehdipour Gianmarco Bernasconi Whye Kit Fong Yiluan Guo Holger Caesar†

***Abstract*— Machine Learning (ML) has replaced handcrafted methods for perception and prediction in autonomous vehicles. Yet for the equally important planning task, the adoption of ML-based techniques is slow. We present nuPlan, the world's first real-world autonomous driving dataset and benchmark. The benchmark is designed to test the ability of ML-based planners to handle diverse driving situations and to make safe and efficient decisions. We introduce a new large-scale dataset that consists of 1282 hours of diverse driving scenarios from 4 cities (Las Vegas, Boston, Pittsburgh, and Singapore) and includes high-quality auto-labeled object tracks and traffic light data. We mine and taxonomize common & rare driving scenarios which are used during evaluation to get fine-grained insights into the performance and characteristics of a planner. Beyond the dataset, we provide a simulation and evaluation framework that enables a planner's actions to be simulated in closed-loop to account for interactions with other traffic participants. We present a detailed analysis of numerous baselines and investigate gaps between ML-based and traditional methods. Find the nuPlan dataset and code at nuplan.org.**

I. Introduction

In the last decade, autonomous vehicle perception and prediction have been revolutionized by deep learning-based methods trained on large-scale datasets [1], [2], [3], [4], [5], [6], [7]. While similar attempts have been made in the field of learning-based or neural planning, these are not yet able to surpass their rule-based counterparts. One possible reason is the difficulty of generalizing driving scenarios when learned from a limited number of examples. Furthermore, driving scenarios typically follow a long-tail distribution, which further exacerbates the generalization issue. Finally, learning-based planning lacks formal safety guarantees, thus making it potentially unsafe and challenging to certify.

We introduce the nuPlan dataset and simulation framework for autonomous vehicle planning. Our goal is to create a testbed for open-loop and closed-loop planning starting in real-world scenarios. This test bed is then used to compare traditional, learning-based, and hybrid planners. nuPlan enables numerous novel types of research, such as learning-based planning, the interplay between prediction and planning, and end-to-end planning using a large amount of published sensor data. We make the following contributions:

- We release the largest dataset for autonomous driving to date, with a total of 1282h from 4 cities. We also publish an unprecedented 128h of sensor data.
- We develop techniques to auto-label the dataset with accurate object tracks, traffic lights, and scenario labels.

Authors are or were (†) with Motional. (*) indicates equal contribution.

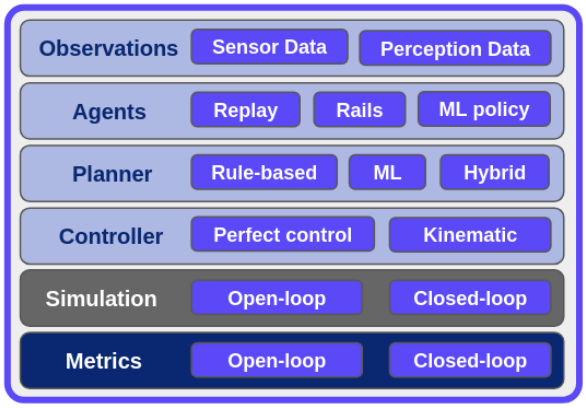

Fig. 1. An overview of the nuPlan simulation framework.

- We publish our closed-loop simulation and evaluation framework (Fig. 1) and compare the performance of traditional and learning-based planners to identify gaps.

II. Related Work

In this section we discuss the most important datasets and simulators for autonomous vehicle planning. Based on these, we introduce the main categories of planners: classical, learning-based, and hybrid planners.

A. Datasets

Tab. I provides an overview of large-scale prediction and planning datasets with more than 20h of data. We omit smaller datasets like Interaction [8], highD [9], inD [10], OpenDD [11] and CommonRoad [12] and subsets of datasets not focused on prediction or planning [13], [14], [15]. With the exception of nuPlan and CommonRoad [12], all datasets focus on prediction (motion forecasting). Offline perception [16] is crucial to train and evaluate planners using high-quality object tracks, but only present in Waymo [17], MONA [18] and nuPlan. Likewise, the availability of traffic light statuses is crucial for realistic traffic simulation, but only Waymo [17] and Lyft [19] contain these and only from an online perception system, rather than developing offline traffic light status inference as in nuPlan. To assess planning performance we need to focus on specific scenarios and evaluate them in closed-loop. nuPlan is the first dataset to feature both scenario tags and a closed-loop simulation framework. Lyft [19] provides interactive tutorials for closed-loop simulation, but they lack the modular framework, evaluation server, and hold-out test set of nuPlan. Finally, we need a large-scale dataset to generalize well. Only the

Lyft [19], Shifts [15] and nuPlan datasets provide more than 1000h of driving data and nuPlan was the first such dataset that provides lidar and camera sensor data (128h), although Waymo [17] later also released compressed lidar data.

B. Simulation

Many works in the literature use proprietary simulators [20], [21]. While sharing their approaches to planning, they do not provide enough information to reproduce their experimental results. Graphical simulators like CARLA [22] and AirSim [23] focus on photorealistic rendering, but lack the realism of real-world maps and agent behavior. CommonRoad [12] was the first open-source simulator focused on planning. However, CommonRoad [12] does not provide a real world dataset as the basis for the simulation, instead resorting to a small number of manually crafted scenarios and tools to import other datasets, albeit without any sensor data. With nuPlan we aim to overcome the above limitations by releasing a large-scale real-world dataset and an open-source closed-loop simulator. Following the release of nuPlan, ScenarioNet [24] focused specifically on Reinforcement Learning and integrated nuPlan and other datasets into their pipeline. They also interfaced with MetaDrive [25] that enables graphical simulation of nuPlan.

C. Planning

Classical planning. The planning problem has long been treated as an optimization problem in the traditional approaches [26], [27], [28], [29], [30], [31]. By carefully designing a cost function, the optimization aims to generate the optimal trajectory that minimizes the cost function in the corresponding search space (e.g., A* search [27], [28], sampling-based methods [32], [33], dynamic programming [30]). While these approaches enjoy the theoretical guarantees on the convergence to an optimal solution, handcrafting the cost function that represents the human-like driving behavior is challenging. In practice, many studies rely on tremendous engineering efforts to fine-tune the solution.

Learning based planning. Pioneering by the study of [34], the idea of using a neural network to imitate expert driver and directly output driving control command provides an alternative planning solution. With the recent success of deep learning, the learning based planning received considerable attention [35], [36], [37], [38], [39], [40], [20], [41].

Imitation learning (IL) and inverse reinforcement learning (IRL): IL trains a model to either map the sensor data directly (end-to-end system), or indirectly through the perception and prediction models (modularized system), to the expert driver actions (e.g. steering and speed profile) [34], [37], [36], [42]. With the advancements in deep learning literature, IL studies adopt the state-of-the-art supervised learning models architectures to learn better scene representations [43], [20]. IL often suffers from a poor generalization where the compounding error leads to driving scenarios that are outside of the training data, known as "covariate shift" [34]. Carefully designed data augmentation is often used to address this issue [36]. As an alternative to directly imitating the driver behavior, IRL aims to learn an unknown reward function that explains expert demonstrations [44]. Once learned, such reward function is used to infer the optimal trajectory from a set of pre-defined or generated trajectories [45]. The maximum entropy formulation of IRL [46] has been applied to autonomous driving where the reward function is estimated based on a set of handcrafted features in [47], [48], [49].

Reinforcement learning (RL): RL learns the optimal driving behavior by interacting with the environment and optimizing a given reward function. RL is well-suited for handling the interaction between the agent and the environment in a sequential decision process. However, due to its learning by "trial-and-error" search nature, studies in RL rely on the driving simulation to provide the environment [50], [51], [52]. These studies have demonstrated strong performance in simulations. The real-world applications of RL in autonomous driving are also reported in [53], [54].

Hybrid solutions. Hybrid solutions are proposed to leverage the advantages of both the classical and learning based planning. Several studies use learning to improve the classical planning algorithm. These include using a learning model to guide the exploration for sampling-based path-planners [55], using a learning model to improve the efficiency of sampling-based motion-planners in high dimensional setting [56], applying optimizer to actively rectifies the learning model's plan to satisfy the safety and comfort requirements [57], [40]. Other studies leverage the classical planning to generate the trajectory candidates, which are passed to an ML-based model to evaluate [49], [38], [58].

System design. While planning is the ultimate goal of the autonomous driving system, different system designs to assemble the perception and prediction introduce opportunities and challenges to improve the planning performance. Most autonomous driving systems use a multi-stage pipeline of independent tasks like perception, prediction and planning [59], [60], [61]. The hope is that the performance gain on individual tasks translates to a better planning performance. In contrast, various studies consider multi-task learning (MTL) where they jointly train models to perform perception, prediction and planning simultaneously [62], [63], [58], [38]. These works have shown that MTL achieves better data utilization at lower computation cost. Recently, some studies leverage the query-based design in transformer architectures to integrate all tasks in a unified framework that is trainable end-to-end [64], [65], [66], [67], [41], [68]. Such a framework encourages better spatio-temporal feature learning from the sensor data and directly improves the planning performance.

III. Dataset

In this section, we describe how we collect the nuPlan dataset. We enhance it by auto-labeling the object tracks of other agents and traffic lights, as well as mining for scenarios that are relevant for tracking.

A. Data collection

We collected data from 4 cities (Boston, Pittsburgh, Las Vegas, and Singapore) to build a benchmark dataset for ML-

TABLE I

AN OVERVIEW OF DATASETS FOR PREDICTION AND PLANNING (* SEE SEC. II-A FOR A DETAILED DISCUSSION)

Dataset	Year	Tasks	Perception	Traffic Lights	Scenario Tags	Closed-loop	Total Volume (h)	Sensor Volume (h)
Argoverse 1 [13]	2019	Pred	online	no	no	no	320	0
Waymo [17]	2019	Pred	**offline**	online	no	no	574	0*
Lyft [19]	2020	Pred	online	online	no	no*	1118	0
Shifts [15]	2021	Pred	n/a	no	no	no	**1667***	0
MONA [18]	2022	Pred	**offline**	no	no	no	130	0
Argoverse 2 [14]	2023	Pred	online	no	no	no	763	0
nuPlan	2021	**Pred+Plan**	**offline**	**offline**	**yes**	**yes**	**1282**	**128**

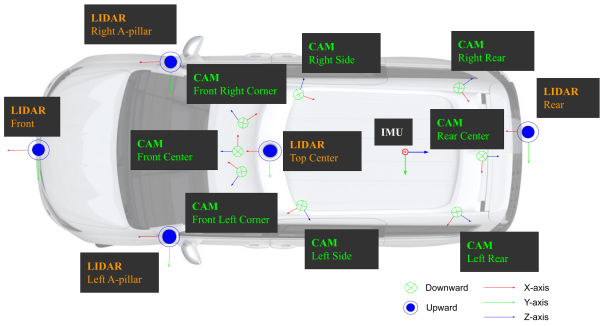

Fig. 2. The data collection vehicle's sensor setup.

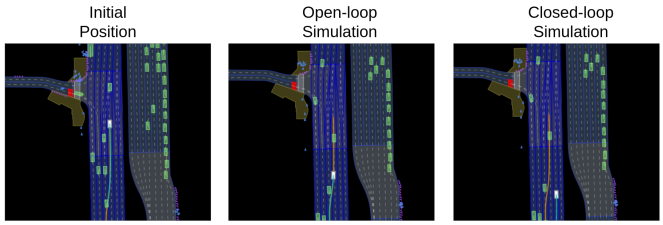

Fig. 3. Open-loop (OL) v.s. closed-loop (CL) simulation. In OL, ego vehicle follows the ground-truth trajectories. In CL, ego vehicle follows the planner model output. As shown above, starting from the initial scenario (left), using a trained machine learning planner (Urban Driver model in Sec. V), we show the different simulation roll-out in open-loop (middle) and closed-loop (right). [Ego ground-truth trajectory, Planner output trajectory]

based planning. In total, we have 1282 hours of challenging and real-world driving scenarios. For example, double parking in Boston, custom precedence patterns for left turns in Pittsburgh, crowded pick-up and drop-off points (PUDOs) in casinos in Las Vegas, and left-hand traffic in Singapore. We exclude heavy rain and night data, as these would impact the quality of our perception system (see Sec. III-B).

Manual driving. We use Chrysler Pacifica Plug-in Hybrid Electric Vehicles (PHEV) to drive in these cities. See Fig. 2 for the sensor setup. Our vehicle operators (VOs) are instructed to use a natural driving style and drive safely. Since our focus is on planning, it is crucial that we drive manually, while most other datasets [13], [14], [17], [19] use a combination of manual and automated driving, which may lead the planner to imitate less desirable driving behavior. The VOs drive from a predefined starting point to a goal using a known route. For example, we drive between various hotels and casinos on the Las Vegas strip, which are typical routes for our robotaxi and are known and mapped beforehand.

Sensor data. Sensor data include lidar point clouds and camera images. Due to the vast scale of the full sensor dataset (200+ TB), we only release a subset of the sensor data which totals 128 hours. This subset was selected to satisfy all stratification constraints as described below.

Maps. Similar to nuScenes [69], nuPlan provides detailed human-annotated 2D high-definition semantic maps of the driving locations. We release rasterized and vectorized maps. While rasterized maps are useful for simplicity and efficient lookup, vectorized maps provide more precise geometric information and metadata. Examples of semantic map layers are lanes, car parks, crosswalks and stop lines (see Fig. 4).

B. Auto-labeling

In order to faithfully reconstruct various driving scenarios, we develop an auto-labeling system. It first generates the tracks for all the objects in the scene; then traffic light statuses are inferred from these tracks. Based on the above labels, we can reliably mine different driving scenarios.

Offline perception. We build an offline perception system to label the objects in the scene [16] automatically. Compared with the online perception systems used in many other datasets [19], [13], the offline version is not constrained by latency and causality. Therefore, the fidelity of the generated tracks is drastically higher, enabling us to evaluate planning performance under very limited perception noise.

Inspired by [16], our offline perception system contains three stages: 3D object detection, offline tracking, and global track refinement. The detector in the first stage takes the point clouds from both the top lidar and side lidars as input and detects the bounding boxes through a large neural network [70]. The offline tracker leverages both past and future detections in an extended time window to generate tracks. In the last step, a novel network is developed to load both tracks and points clouds within the tracks to refine the attributes of all the bounding boxes of vehicle class, such as positions, headings, sizes and velocity.

Traffic light status. To create a realistic simulation of the environment, it is crucial to capture the traffic light statuses. Existing datasets lack traffic lights [13], [14] or use online vision-based systems to detect their statuses [17], [19]. In contrast, we develop a novel offline system to automatically label the statuses of traffic lights by inferring them from the motion of the actors present in the scene. Our labeling system is able to cover all lanes with observed agents.

We make use of the detections and tracks produced by our offline perception system, as well as map information. To infer a green traffic light status within a given intersection, we determine if there are agents moving within the intersection in the direction controlled by the particular traffic light. To infer a red traffic light status, we check for agents slowing

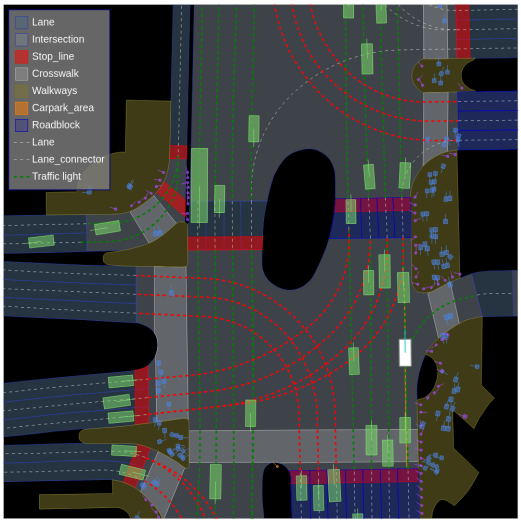

Fig. 4. Semantic map of nuPlan with 10 semantic layers in different colors, polygons and lines. It also includes traffic light statuses encoded into the lane connectors.

down or being stationary in the lane that approaches the intersection and controlled by the particular traffic light.

Scenario mining. Traditional approaches to evaluate planning performance are dominated by monotonous lane following scenarios. To get a fine-grained understanding of the planning performance, we develop a scenario taxonomy and scenario mining algorithms using low-level attributes like vehicle speed and state transitions. The attributes can be inferred from offline perception tracks and traffic light statuses. In total, we have 73 unique scenario types.

IV. Simulation

nuPlan provides a simulation framework (Fig. 1) that is modular and flexible to work with different datasets and setups. The simulation is initialized with the real-world *observations* captured in the dataset, namely raw sensor data or object tracks. Given these environment observations, an *agent* model can be used to predict the future trajectories of all agents. Observations and agent trajectories are passed to a *planner* that predicts the best route for the ego vehicle given the other agents' routes. Finally, a *controller* converts the intended route into a feasible trajectory. The simulation can either playback the actions recorded in the dataset (open-loop) or allow the simulation to deviate from the recording by incorporating the ego's actions (closed-loop). Below are the simulation components in detail.

A. Agents

Of the 6 object classes, vehicle, pedestrian, generic object, traffic cone, barrier, bicycle, construction zone sign, 3 are moving object classes simulated as agents: vehicles, pedestrians, and cyclists. Agents are dynamic objects that can move as the scenario evolves.

A well-known method is to simply propagate agents according to the logged data. We refer to these as non-reactive log-replay agents. Log-replay agents are used to simulate scenarios both in open-loop and closed-loop, a near-perfect recreation of the recorded data. However, closed-loop simulation quickly diverges if the planner decides to take different actions from what is recorded in the log. Thus, in closed-loop simulation agents can be also simulated with the aim to interact with the ego vehicle and with each other by producing novel simulation states that resemble real agent behaviors. We refer to these as reactive agents. Reactive agents are only relevant in closed-loop simulation and by definition open-loop simulation uses non-reactive agents.

We develop reactive agents following the Intelligent Driver Model [71] (IDM) policy. IDM agents are initialized with the initial pose and velocity of the logged agents. The agents follow the lane center line of the underlying map. The longitudinal control is dictated by the IDM policy. This allows the agents to react to the ego's actions, as well as other reactive agents. In turn, this reduces false collisions and lets scenarios play out for longer. Note that we only apply this policy to vehicles, while Vulnerable Road Users (VRUs) are replayed from the log. We choose not to model VRUs as reactive agents as their behavior is often uncooperative and thus hard to model.

B. Controller

In nuPlan, planners provide a trajectory as a sequence of poses in $SE(2)$, without any kinematic feasibility requirement. This trajectory is assumed to be sampled at specific times in the future according to the simulation configuration. To assert kinematic feasibility and prevent users from cheating, we require the use of a controller. nuPlan provides the flexibility to use any controller, such as perfect tracking, which simply interpolates poses along the planned trajectory.

We developed a two-stage controller to propagate the simulation in closed-loop. This controller consists of two parts, a trajectory tracker and a motion model to forward-integrate the simulation. We implement a Linear Quadratic Regulator (LQR) [72], [73] as the tracker. The found optimal control policy is then fed to the second part of the simulation controller, a kinematic bicycle model which is forward-integrated to propagate the simulation state. Alternatively, we also support different trackers in the two-stage controller, such as an iterative-LQR tracker.

C. Evaluation

Different metrics and frameworks have been explored for scoring models in prediction and motion planning benchmarks [17], [74], [75], [76], [77], [78], [79], [80]. In this paper, we select a set of metrics and design an aggregation method to compare the performance of planners. In open-loop, we only evaluate the closeness of the planner generated trajectory to the human-driven trajectory. The open-loop metrics are modified Average Displacement Error (ADE), Final Displacement Error (FDE), Average Heading Error (AHE), Final Heading Error (FHE), and Miss Rate (MR) in which we calculate the metrics over different horizons and report their average score. For closed-loop, we use a combination of metrics to evaluate lawfulness and compliance with traffic rules consisting of no at-fault collisions, trajectories inside drivable area, no trajectories in lanes belonging to oncoming

TABLE II

PLANNER METRICS THEIR AND WEIGHTS

Simulation	Metric name	Multiplier weight	Average weight
Open-loop	MR within bound	{0, 1}	-
	AHE and FHE within bound	-	2
	ADE and FDE within bound	-	1
Closed-loop	No at-fault collisions	{0, 0.5, 1}	-
	Drivable area compliance	{0, 1}	-
	Making progress	{0, 1}	-
	Driving direction compliance	{0, 0.5, 1}	-
	TTC within bound	-	5
	Progress along route ratio	-	5
	Speed limit compliance	-	4
	Comfort	-	2

traffic, not driving above the speed limit and maintaining enough Time To Collision (TTC) with other road users, metrics to evaluate progress towards the goal and measure the rider comfort. All metrics' scores are normalized to the range $[0 - 1]$ using thresholds that are selected based on legal requirements and natural human driving. A higher score indicates a better performance.

The final score of a planner is computed by averaging the scores for its generated trajectories across all scenarios. The score of a trajectory in a scenario is given by a hybrid weighted average of all metrics' scores.

The rest of the metrics are weighted according to their importance (See Tab. II) and then averaged to compute the scenario score as:

$$\textit{scenario score} = \prod_{i \in \text{multiplier metrics}} score_i \times \sum_{j \in \text{average metrics}} weight_j \times score_j$$

We define the score for each challenge (open-loop, closed-loop non-reactive, and closed-loop reactive) as the average scenario score across all scenarios for that challenge.

V. EXPERIMENTS

Here we present a number of planning baselines and their results when evaluated on the nuPlan benchmark. We analyze how the planning performance is impacted by lower quality perception inputs, as well as how it generalizes to other cities. Finally, we discuss the new state-of-the-art set by the submissions to the first nuPlan challenge.

A. Planning baselines

We implement several planning methods that are representative of the literature.

a) Simple Planner: The Simple planner has little planning capability. The planner plans a straight line at a constant speed. The only logic of this planner is to decelerate if the current velocity exceeds the max velocity.

b) IDM Planner: The Intelligent Driver Model (IDM) planner is essentially an Adaptive Cruise Control (ACC) policy [81]. The planner consists of two parts: path planning and longitudinal control. The path planning component is a breadth-first search algorithm. It finds a center-line path toward the mission goal extracted from the underlying map structure. The longitudinal control follows the IDM policy. The policy describes how fast the planner should go based on the distance between itself and the closest leading agent.

TABLE III

MAIN RESULTS

Planner	Open-loop	Closed-loop Non-reactive	Closed-loop Reactive
Simple Planner	0.22	0.32	0.37
IDM Planner	0.30	**0.73**	**0.76**
Raster Planner	0.52	0.47	0.46
UrbanDriver	**0.90**	0.68	0.67

c) Raster ML planner: Similar to the encoder in [36], [82], the raster planner uses ResNet-50 [83] as the backbone to encode features from an ego-centric multi-channel raster representing the ego, the agents and the map. The model directly outputs the final ego trajectory. The planner does not perform any post-processing on the predicted ego trajectory.

d) UrbanDriver ML Planner: We adopted an open-loop training variant of the UrbanDriver model [84] as a representative machine learning planner baseline. The model processes vectorized agents and map inputs into local feature descriptors that are passed to a global attention mechanism for yielding a predicted ego trajectory. We train the model using imitation learning to match expert trajectories available in the nuPlan dataset. Data augmentation is additionally performed on the agents and expert trajectory provided during training to mitigate data distribution drift encountered during closed-loop simulation. This version was used for the challenge. We also implemented a multi-step prediction baseline variant as discussed in [84] and originally proposed in [85] to further address the distribution shift, for the experiments in this work but do not open-source this implementation.

B. Main results

Tab. III shows the planning results for the proposed baselines in each of the three challenge setups. Supervised learning-based planners excel in an open-loop setting. This is unsurprising as the task is akin to the traditional motion forecasting challenge. This suggests that an ML planner can choose to make similar decisions to a human driver in open-loop settings. However, ML planners still struggle to overcome the distribution shift in closed-loop. A closed-loop scenario can develop into a new situation that was never present in the training dataset. Even techniques such as data augmentation and closed-loop training fail to overcome this domain gap. This is evident in both the literature [36] and our experiments. Rule-based planners, on the other hand, face no such issues. Policies like IDM can produce decent driving behavior. This is confirmed by the metrics as it achieved the highest scores for closed-loop. It should be noted though that the reactive agents are also modelled with a similar IDM. The use of similar assumptions on the vehicle behavior may result in giving the IDM planner an unfair advantage over other planners in closed-loop evaluation. It is evident that sufficiently sophisticated rule-based planners still outperform purely learned planners in closed-loop settings.

C. Perturbation

As the nuPlan dataset is created with offline perception, to capture the original probability distribution of data collected online, we injected uniform noise on the detections. Noise

TABLE IV

DETECTION PERTURBATION IN CLOSED-LOOP REACTIVE SIMULATION

Simulation Agents	UrbanDriver	UrbanDriverOnline
Original	**0.67**	**0.60**
Perturbed	0.64	0.59

TABLE V

URBANDRIVER LOCATION GENERALIZATION

Locations	Open-Loop	Closed-Loop Non-reactive	Closed-Loop Reactive
Las Vegas	**0.91**	**0.57**	**0.57**
Singapore	0.28	0.23	0.18
Boston	0.57	0.49	0.50
Pittsburgh	0.41	0.39	0.32

was added in the dimensions and the pose of the detected agents, with variance extracted by comparing offline and online detections. The scores of planners under nominal and noise-injected simulations in closed-loop reactive mode are presented in Tab. IV. A version of UrbanDriver trained on the perturbed data is called UrbanDriverOnline, which shows a performance deterioration compared to the nominal model on both nominal and injected data. This indicates the value of high-quality offline annotations in the dataset and the learning pipeline.

D. Generalization

The location generalization experiment is shown in Tab. V. The experiment aims to test a model's generalization capabilities. The UrbanDriver model was trained purely on data from Las Vegas. The model was tested separately on scenarios from Singapore, Boston, Pittsburgh, and Las Vegas. The open-loop performance dropped by 53.8%, while closed-loop non-reactive and closed-loop reactive performance dropped by 35.1% and 41.5% respectively. The worst-performing location is Singapore. This can be explained by the left-hand traffic, while the model was trained on right-hand traffic. One insight is that the correlation between the model's open-loop and closed-loop performance is relatively weak. The difference between open and closed-loop scores across Singapore, Boston, and Pittsburgh is only 16.3%, while for Las Vegas it is more than double at 37.3%. This indicates that a good motion forecasting model does not translate to closed-loop capabilities. Thus a major challenge is to overcome the domain gap between open-loop and closed-loop before tackling larger generalization problems.

E. nuPlan challenge

In the nuPlan motion planning challenge contestants create a planner to traverse a set of diverse and challenging scenarios across all four cities. Tab. VI shows the Overall Score, which is the average across Open-loop, Closed-loop Non-reactive, and Closed-loop Reactive challenges of the top four planners. In the open-loop challenge, planners that incorporated supervised learned methods scored relatively well. In the closed-loop challenges, planners employed a combination of learned and handcrafted components. A common theme was the use of a learned model to first predict the ego's planned trajectory. [82] uses a raster-based model

TABLE VI

NUPLAN CHALLENGE LEADERBOARD

Team name	Approach	Overall Score
CS Tu [87]	Rule-based + ML refinement	**0.895**
AutoHorizon [82]	ML + optimizer	0.875
Pegasus	ML + collision checking	0.848
AID [86]	ML + hierarchical game theory	0.829

that outputs a spatial-temporal heatmap for the ego and an occupancy map for the surrounding agents. [86] are vector-based using transformers as a backbone. Once the trajectory is obtained, the planner has a further refinement stage used to ensure kinematic feasibility and collision avoidance. The highest-scoring planner in closed-loop was mostly rule-based [87]. It generates a handful of trajectories by perturbing the center line laterally at different velocities. Trajectories are selected with a heuristic that considers factors such as collision, drivable areas, traffic laws, and comfort. An ML-generated trajectory is fused to correct the long-term planned horizon. This limited the influence of the learned model. We draw two conclusions from the challenge results. First, ML-based methods require additional post-processing for closed-loop driving. Second, hybrid methods appear to be the most effective approach, combining traditional and data-driven methods.

VI. CONCLUSION

We presented nuPlan, the first real-world driving benchmark and the largest existing labeled autonomous driving dataset. The dataset consists of 1282 hours of diverse driving scenarios across 4 cities as well as an unprecedented 128 hours of raw sensor data and is accompanied by an evaluation framework powered by a closed-loop simulator; the dataset and the evaluation framework are publicly available. We investigated the state of current rule-based and learned-based planners by evaluating multiple approaches on the nuPlan dataset across challenging driving scenarios. The first public nuPlan challenge demonstrated that rule-based planners outperform purely ML-based ones, but hybrid planners with learned-based components show the most promise in handling difficult scenarios. In the future, we plan to mine for richer long-tail driving scenarios, design scenario-based metrics, provide ML-based planning and agent baselines and explore end-to-end planner training directly from sensor data.

ACKNOWLEDGEMENT

We would like to thank the numerous current and former colleagues at Motional who contributed to this work, especially Juraj Kabzan, Edouard Capellier, Abhinav Rai, Xiaoli Meng, Jiong Yang, Lubing Zhou, Abirami Srinivasan, Bing Jui Ho, Hiok Hian Ong, Mitchell Spryn, Taufik Tirtosudiro, Michael Noronha, Vijay Govindarajan, Jeff Wang, Nelly Lyu, Samson Hang, Shashank Chaudhary, Patrick Weygand, Vern Jensen, Abhimanyu Singh, Qiang Xu and Sammy Omari. Furthermore, many external advisors gave useful feedback, including Kashyap Chitta and Oscar de Groot.

REFERENCES

[1] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *CVPR*, 2019.

[3] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *CVPR*, 2020.

[4] Q. Chen, S. Vora, and O. Beijbom, "Polarstream: Streaming object detection and segmentation with polar pillars," in *NeurIPS*, 2021.

[5] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," in *CVPR*, 2021.

[6] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *CVPR*, 2020.

[7] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

[8] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," *arXiv preprint arXiv:1910.03088*, 2019.

[9] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *ITSC*, 2018.

[10] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *IV*, 2020.

[11] A. Breuer, J.-A. Termöhlen, S. Homoceanu, and T. Fingscheidt, "openDD: A large-scale roundabout drone dataset," in *ITSC*, 2020.

[12] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *IV*, 2017.

[13] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019.

[14] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *NeurIPS Track on Datasets and Benchmarks*, 2021.

[15] A. Malinin, N. Band, Y. Gal, M. Gales, A. Ganshin, G. Chesnokov, A. Noskov, A. Ploskonosov, L. Prokhorenkova, I. Provilkov, V. Raina, V. Raina, D. Roginskiy, M. Shmatova, P. Tigas, and B. Yangel, "Shifts: A dataset of real distributional shift across multiple large-scale tasks," in *NeurIPS Track on Datasets and Benchmarks*, 2021.

[16] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *CVPR*, 2021.

[17] S. Ettinger, S. Cheng, and B. C. et al., "Large scale interactive motion forecasting for autonomous driving: The Waymo Open Motion Dataset," in *ICCV*, 2021.

[18] L. Gressenbuch, K. Esterle, T. Kessler, and M. Althoff, "Mona: The munich motion dataset of natural driving," in *ITSC*, 2022.

[19] J. Houston, G. Zuidhof, and L. B. et al., "One thousand and one hours: Self-driving motion prediction dataset," in *CoRL*, 2020.

[20] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *CoRL*, 2022.

[21] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *RSS*, 2019.

[22] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," *CoRR*, 2017.

[23] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*, 2018.

[24] Z. A. Daniels and D. Metaxas, "Scenarionet: An interpretable data-driven model for scene understanding," in *IJCAI Workshop on Explainable Artificial Intelligence (XAI) 2018*, 2018.

[25] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *PAMI*, 2022.

[26] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, 2016.

[27] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*. Springer, 2009, vol. 56.

[28] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *IROS*, 2018.

[29] T. Fraichard and C. Laugier, "Path-velocity decomposition revisited and applied to dynamic trajectory planning," in *ICRA*, 1993.

[30] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo EM motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[31] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, 2008.

[32] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, 2000.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, 2011.

[34] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NeurIPS*, 1988.

[35] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *ICCV*, 2019.

[36] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *RSS*, 2019.

[37] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[38] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *CVPR*, 2019.

[39] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah *et al.*, "Urban driving with conditional imitation learning," in *ICRA*, 2020.

[40] M. Vitelli, Y. Chang, Y. Ye, A. Ferreira, M. Wołczyk, B. Osiński, M. Niendorf, H. Grimmett, Q. Huang, A. Jain *et al.*, "Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies," in *ICRA*, 2022.

[41] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *CVPR*, 2023.

[42] M. Hallgarten, M. Stoll, and A. Zell, "From prediction to planning with goal conditioned lane graph traversals," *arXiv preprint arXiv:2302.07753*, 2023.

[43] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," in *CoRL*, 2018.

[44] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[45] J. Ho and S. Ermon, "Generative adversarial imitation learning," *NIPS*, 2016.

[46] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *IROS*, 2011.

[47] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[48] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *AAAI*, 2008.

[49] T. Phan-Minh, F. Howington, T.-S. Chu, S. U. Lee, M. S. Tomov, N. Li, C. Dicle, S. Findler, F. Suarez-Ruiz, R. Beaudoin *et al.*, "DriveIRL: Drive in real life with inverse reinforcement learning," in *ICRA*, 2023.

[50] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[51] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *CoRL*, 2020.

[52] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *ICCV*, 2021.

[53] M. Riedmiller, M. Montemerlo, and H. Dahlkamp, "Learning to drive a real car in 20 minutes," in *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, 2007.

[54] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *ICRA*, 2019.

[55] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms," in *IROS*, 2015.

[56] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *ICRA*, 2019.

[57] H. Pulver, F. Eiras, L. Carozza, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "Pilot: Efficient planning by imitation learning and optimisation for safe autonomous driving," in *IROS*, 2021.

[58] S. Casas, A. Sadat, and R. Urtasun, "Mp3: A unified model to map, perceive, predict and plan," in *CVPR*, 2021.

[59] L. Chen, L. Platinsky, S. Speichert, B. Osiński, O. Scheel, Y. Ye, H. Grimmett, L. Del Pero, and P. Ondruska, "What data do we need for training an av motion planner?" in *ICRA*, 2021.

[60] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.

[61] K. Xiong, S. Gong, X. Ye, X. Tan, J. Wan, E. Ding, J. Wang, and X. Bai, "Cape: Camera view position embedding for multi-view 3d object detection," in *CVPR*, 2023.

[62] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *ECCV*, 2020.

[63] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal *et al.*, "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *ICLR*, 2022.

[64] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *CVPR*, 2022.

[65] S. Casas, A. Sadat, and R. Urtasun, "MP3: A unified model to map, perceive, predict and plan," in *CVPR*, 2021.

[66] K. Chitta, A. Prakash, and A. Geiger, "Neat: Neural attention fields for end-to-end autonomous driving," in *ICCV*, 2021.

[67] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao, "St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning," in *ECCV*, 2022.

[68] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," in *NeurIPS*, 2022.

[69] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[70] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *CVPR*, 2020.

[71] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, 2000.

[72] J. Liu, Z. Yang, Z. Huang, W. Li, S. Dang, and H. Li, "Simulation performance evaluation of pure pursuit, stanley, lqr, mpc controller for autonomous vehicles," in *2021 IEEE international conference on real-time computing and robotics (RCAR)*, 2021.

[73] B. Varma, N. Swamy, and S. Mukherjee, "Trajectory tracking of autonomous vehicles using different control techniques (pid vs lqr vs mpc)," in *2020 International conference on smart technologies in computing, electrical and electronics (ICSTCEE)*, 2020.

[74] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos, "Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic," in *International Conference on Formal Methods and Models for System Design*, 2019.

[75] W. Xiao, N. Mehdipour, A. Collin, A. Y. Bin-Nun, E. Frazzoli, R. D. Tebbens, and C. Belta, "Rule-based optimal control for autonomous driving," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, 2021, pp. 143–154.

[76] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," in *CoRL*, 2021.

[77] R. McAllister, B. Wulfe, J. Mercat, L. Ellis, S. Levine, and A. Gaidon, "Control-aware prediction objectives for autonomous driving," in *ICRA*, 2022.

[78] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *IV*, 2022.

[79] S. Jiang, Z. Xiong, W. Lin, Y. Cao, Z. Xia, J. Miao, and Q. Luo, "An efficient framework for reliable and personalized motion planner in autonomous driving," *RA-L*, 2022.

[80] M. Ilievski, "Wisebench: A motion planning benchmarking framework for autonomous vehicles," Master's thesis, University of Waterloo, 2020.

[81] O. Derbel, T. Peter, H. Zebiri, B. Mourllion, and M. Basset, "Modified intelligent driver model for driver safety and traffic stability improvement," *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 744–749, 2013.

[82] Y. Hu, K. Li, P. Liang, J. Qian, Z. Yang, H. Zhang, W. Shao, Z. Ding, W. Xu, and Q. Liu, "Imitation with spatial-temporal heatmap: 2nd place solution for nuplan challenge," 2023.

[83] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019.

[84] O. Scheel, L. Bergamini, M. Wolczyk, B. Osinski, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *CoRL*, 2021.

[85] A. Venkatraman, M. Hebert, and J. Bagnell, "Improving multi-step prediction of learned time series models," *AAAI*, 2015.

[86] Z. Huang, H. Liu, and C. Lv, "Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving," 2023.

[87] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," 2023.