Planning with Adaptive World Models for Autonomous Driving

Arun Balajee Vasudevan, Neehar Peri, Jeff Schneider, Deva Ramanan
Carnegie Mellon University

Abstract: Motion planning is crucial for safe navigation in complex urban environments. Historically, motion planners (MPs) have been evaluated with procedurally-generated simulators like CARLA. However, such synthetic benchmarks do not capture real-world multi-agent interactions. nuPlan, a recently released MP benchmark, addresses this limitation by augmenting real-world driving logs with closed-loop simulation logic, effectively turning the fixed dataset into a reactive simulator. We analyze the characteristics of nuPlan's recorded logs and find that each city has its own unique driving behaviors, suggesting that robust planners must adapt to different environments. We learn to model such unique behaviors with BehaviorNet, a graph convolutional neural network (GCNN) that predicts *reactive* agent behaviors using features derived from recently-observed agent histories; intuitively, some aggressive agents may tailgate lead vehicles, while others may not. To model such phenomena, BehaviorNet predicts parameters of an agent's motion controller rather than predicting its spacetime trajectory (as most forecasters do). Finally, we present AdaptiveDriver, a model-predictive control (MPC) based planner that unrolls different world models conditioned on BehaviorNet's predictions. Our extensive experiments demonstrate that AdaptiveDriver achieves state-of-the-art results on the nuPlan closed-loop planning benchmark, improving over prior work by 2% on Test-14 Hard R-CLS.

Keywords: Motion Planning, Autonomous Vehicles, Data-Driven Simulation

1 Introduction

Motion planning (MP) is a critical component of the autonomy stack. Autonomous Vehicles (AVs) must carefully plan their motion to navigate in complex urban environments to safely reach their goal, avoid collisions, and abide by the rules of the road. Motion planners are typically trained and evaluated in synthetic environments like CARLA [1] and AirSim [2]. However, such simulated environments notoriously suffer from a sim-to-real gap due to systematic biases and a lack of real-world diversity. Instead, we focus on the recent nuPlan planning benchmark [3], which makes use of *real* driving logs to create environments for training and evaluation. Data-driven simulators like nuPlan are a core enabling technology in current autonomy stacks through the use of *re*simulation and log playback.

Planning Evaluation. Although nuPlan evaluates algorithms on ego-centric forecasting accuracy (defined as the OLS metric [3]) and motion planning in a non-reactive world (NR-CLS), our work primarily focuses on motion planning in a reactive world (R-CLS) as this most closely resembles real-world deployment. Recent methods like PDM-C [4] have dramatically improved planning performance in reactive environments by combining model-predictive control (MPC) with classical motion controllers like the Intelligent Driver Model (IDM) [5].

Model Predictive Control. Surprisingly, rule-based methods like PDM-C still outperform learning-based planners on real data. PDM-C builds upon IDM by incorporating several key ideas from the MPC literature like proposal generation, world model rollouts, and proposal scoring. First, PDM-C

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

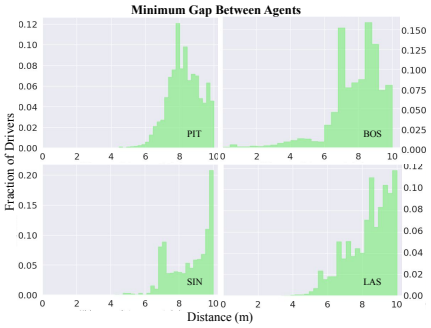

Figure 1: **Drivers Behave Differently in Different Cities.** Using the nuPlan benchmark, we compare the distance between the ego-vehicle and lead agent (min-gap) in Pittsburgh (PIT), Boston (BOS), Las Vegas (LAS) and Singapore (SIN). Interestingly, we find that the min-gap distribution between cities differs dramatically. For example, Boston drivers are more aggressive (e.g. drive with lower average min-gap) than Pittsburgh drivers. This suggests that (a) robust planners must adapt to diverse driving conditions and (b) model-predictive control (MPC) planners may benefit from adaptive world models that capture such behaviors.

identifies the shortest spatial trajectory to the goal using a lane graph-based search. Next, it generates spatio-temporal proposals by applying IDM policies at different target speeds and lateral center-line offsets. Each proposal is evaluated in an internal world model and is scored based on traffic-rule compliance, progress towards the goal, and ego-vehicle comfort. The proposal with the highest score is selected and executed as the final motion plan.

Different World Models for Each City. Intuitively, PDM-C could achieve perfect planning performance given an accurate world model of future agent actions and an infinite number of trajectory proposals. However, evaluating complex world models can be computationally challenging. Instead, PDM-C makes use of a simpler "world-on-rails" model where agents move with constant velocity during rollouts. Notably, this non-reactive model does not accurately simulate agent behaviors, leading to poorly scored proposals. To improve this "world-on-rails" model, we analyze the characteristics of nuPlan's recorded logs and find that each city has its own unique driving behaviors. For example, Boston drivers are more aggressive (e.g. drive with lower average min-gap) than Pittsburgh drivers (cf. Fig. 1). We model such unique behaviors with BehaviorNet, a graph convolution-based network that predicts future agent actions based on features derived from recently observed agent behaviors. Unlike traditional forecasters, BehaviorNet parameterizes future agent behaviors with IDM controls to create a *reactive* world model. Conceptually, each set of predicted IDM control parameters unrolls a different world model.

Building an Adaptive Planner. Finally, we introduce AdaptiveDriver, an MPC-based planner that scores spatio-temporal proposals conditioned on BehaviorNet's reactive world model. Extensive experiments demonstrate that our AdaptiveDriver planner achieves state-of-the-art results on the nuPlan closed-loop planning benchmark, improving over prior work by 2% on Test-14 Hard. Importantly, the novelty of our work is in our parameterization of future agent behaviors as IDM control parameters (rather than non-reactive spacetime forecasts), which implicitly encodes an ensemble of reactive world models that generalize to never-before-seen environments.

Contributions. We present three major contributions.

1. We demonstrate that each city has its own unique driving behaviors, and adapting to these different environments leads to notable improvements in planning performance. Moreover, we find that behaviors can even vary within a city, motivating our next contribution.
2. We propose BehaviorNet, a graph convolutional neural network (GCNN) that predicts driving behaviors parameterized as IDM controls, using features derived from recently observed agents in the surrounding scene.
3. We present AdaptiveDriver, a planner based on model-predictive control (MPC) that unrolls and executes adaptive world models to safely navigate in diverse environments, achieving state-of-the-art closed-loop planning performance on nuPlan.

2 Related Works

Rule-Based Planning. Although recent works focus on learning robust policies by predicting goal-conditioned way-points [6], cost-volumes, and reward functions [7], rule-based planners still

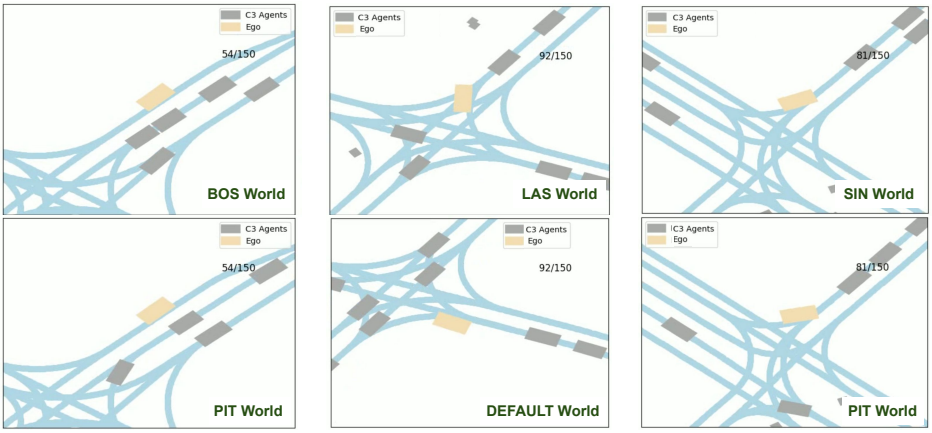

Figure 2: **Visualizing Adaptive World Models.** Each column visualizes the same initial traffic scenario unrolled with different world models (trained on agents from different cities). We visualize the ego-vehicle in yellow and other agents in gray. Blue lines represent the lane-graph. On the **left**, the BOS world model (top) produces agents that tend to tailgate more than the PIT world model, consistent with the min-gap statistics across those two datasets in Figure 1. In the **center**, agents in the LAS world model have a higher max acceleration compared to those of the default IDM world model. On the **right**, agents in the SIN world model have a higher max acceleration and min-gap compared to the PIT world model. We demonstrate that such adaptive world models can be used to significantly improve the accuracy of model-predictive control (MPC) planners. To do so, we train BehaviorNet, a network that predicts parameters of an adaptive world model based on the recent observed history of agents in the scene.

outperform learning-based approaches on real data [4]. Rule-based planners are well studied [8, 9, 10, 11, 12], and have been widely adopted due to their safety guarantees and interpretability [13, 14, 15, 16, 17]. Given the current position, velocity, and distance to the lead vehicle, rule-based planners estimate longitudinal acceleration to safely progress towards the target. The Intelligent Driver Model (IDM) [5] is a classic non-learned algorithm for vehicle motion planning that relies on graph-based search to reach the target while employing a PID velocity controller to avoid collisions with other vehicles. Dauner et. al. [4] upgrades IDM by sampling multiple trajectories and unrolling a constant velocity world model to select an optimal trajectory with minimum cost. This improves collision avoidance without resulting in overly conservative motion plans.

Trajectory Optimization. Motion planning is often framed as an optimization problem of hand-designed cost functions, which are then minimized to generate an optimal trajectory [18, 19, 20, 21]. To simplify this process, cost functions assume a quadratic objective function or divide the planning task into its lateral and longitudinal components. Approaches such as A* [22], RRT [23], and dynamic programming [20] are commonly used to search for optimal solutions. CoverNet [24] generates a set of trajectories and evaluates them based on cost functions, selecting the trajectory with lowest cost. While these methods are attractive due to their parallelizability, interpretability and functional guarantees, they are not robust when applied to real-world scenarios and require significant hyperparameter tuning. Conventional trajectory optimization approaches typically aim to compute a complete trajectory that spans from the initial configuration to the desired goal configuration. However, given the inherently dynamic and uncertain nature of the driving environment, precise long-horizon motion plans cannot be predicted in advance. As a result, model-predictive control (MPC) has gained prominence in recent years for real-time path planning [25, 26, 27, 28] because it adopts an iterative cost minimization strategy to select a locally optimal trajectory for each timestep. This allows MPC-based algorithms to quickly adapt to changes in the environment.

Data-Driven Simulation. In recent years, many learning-based planners have emerged, leveraging the availability of simulator environments like CARLA [1], AirSim [2], and others. However, current simulators are limited because they rely on synthetic data generated from game engines and have insufficient visual fidelity. Importantly, they lack the necessary diversity of driving scenarios required for comprehensive training and evaluation. To address these limitations, [29] proposes multi-agent

City	Target Vel. (θ_0)	Min Gap (θ_1)	Headway Time (θ_2)	Max Acceleration (θ_3)	Max Deceleration (θ_4)
Default IDM World (θ^{IDM})	10	1.0	1.5	1.0	2.0
PIT World (θ^{PIT})	10	2.0	0.5	1.5	3.0
BOS World (θ^{BOS})	10	1.0	1.5	2.0	3.5
SIN World (θ^{SIN})	10	2.5	1.5	2.0	1.0
LAS World (θ^{LAS})	10	1.0	0.5	1.5	0.5

Table 1: **Optimized City-Specific Reactive World Models.** We generate city-specific IDM parameters $(\theta_0, \ldots, \theta_4)$ by optimizing the objective function described in Equation 1 over all training logs from a given city. The optimized world model parameters attempt to minimize the distribution shift between simulated trajectory rollouts and recorded logs while maximizing nuPlan's R-CLS metric on the training set.

behavior models for generating diverse and realistic traffic simulations. More recently, [30] introduced the Waymo sim agents challenge which evaluates simulators by comparing the trajectories of all simulated agents against their ground-truth trajectories. Furthermore, CommonRoad [31] offers a driving dataset and planning benchmark which combines real-world data and rule-based heuristics. In contrast, nuPlan [3] augments real-world driving logs with closed-loop simulation logic, effectively turning the fixed dataset into a reactive simulator. nuPlan has released 1300 hours of real world driving logs from various cities including Las Vegas, Boston, Pittsburgh, and Singapore. Driving in each city presents a unique set of driving challenges. For example, Las Vegas has many high density pick-up and drop-off locations, and intersections with 8 parallel driving lanes per direction. In Boston, drivers tend to double park, creating distinct planning challenges.

3 Planning With An Ensemble of World Models

In this section, we analyze the limitations of PDM-C [4] and propose AdaptiveDriver, an alternative instantiation of model-predictive control that achieves state-of-the-art closed-loop planning performance on the nuPlan benchmark [3].

nuPlan Evaluates Planners in a Reactive Simulator. nuPlan augments real-world driving logs with closed-loop simulation logic, allowing other agents to react to the ego-vehicle. Agents are instantiated with an initial velocity based on their trajectory history and will re-simulate their spatial trajectory from the recorded driving log. The closed-world simulation logic for all agents is initialized with a fixed target velocity (θ_0), minimum gap (θ_1), headway time (θ_2), maximum acceleration (θ_3), and maximum deceleration (θ_4).

Understanding PDM-C's Limitations. PDM-C is a state-of-the-art rule-based planner that improves upon the Intelligent Driver Model (IDM), a car following algorithm that employs a simple longitudinal PID velocity controller along a reference path. PDM-C upgrades IDM to an MPC-based planner by modulating IDM's reference path with different longitudinal velocities and lateral offsets to generate candidate trajectories, internally unrolling a world model for other agents, and selecting the trajectory that minimizes a cost function over that world model. Notably, PDM-C makes use of a simpler "world-on-rails" internal world model where other agents are non-reactive and move with constant velocity during rollouts. Although a 'world-on-rails" model with constant velocity forecasts may work well for short-horizon forecasting, it fails to correctly simulate multi-agent interactions like lane changes, lane merges and stopping at traffic lights.

Predicting Future Agent Actions with BehaviorNet. We improve upon the "world-on-rails" model by learning to predict future agent behaviors with BehaviorNet. We model the unique driving characteristics of each scenario by encoding a vectorized road graph of radius R around the ego-vehicle and two seconds of trajectory history for all nearby agents. BehaviorNet consists of several multi-scale graph convolution and attention modules followed by a fully connected layer to predict IDM control parameters. We further describe BehaviorNet's architecture in the supplement. Notably, unlike traditional forecasters, BehaviorNet directly predicts IDM control parameters $(\theta_0, \ldots, \theta_4)$, which can then used to unroll a reactive world model.

Learning Adaptive Behavior Parameters. We train BehaviorNet with paired examples of past agent trajectories and target IDM control parameters that best explain future agent actions. We

Figure 3: **Visualizing Clusters of Behaviors.** In (a), we optimize scenario-specific world models using Equation 1 and visualize per-log IDM parameters using a tSNE plot, coloring cities differently. For comparison, we also plot per-city-optimized IDM parameters (Table 1) as large colored dots. In (b), we cluster per-scenario IDM parameters using K-means and visualize different clusters. Each cluster represents a unique emergent driving behavior. In (c), we compare the distribution of min-gap between the "aggressive driver" cluster (left) and "passive driver" cluster (right). We note that the "aggressive driver" cluster has a lower average min-gap than the "passive driver" cluster, validating that our K-means clusters encode unique city-agnostic driving behaviors.

optimize target IDM parameters by fitting to training logs using a grid search over $\boldsymbol{\theta} = \{\theta_0, \ldots, \theta_4\}$

$$\boldsymbol{\theta}^{TARGET} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} ||X_{SIM}(\boldsymbol{\theta}) - X_{LOG}||_2^2 \tag{1}$$

where X_{SIM} is the position of all agents in the simulated IDM rollout and X_{LOG} is the position of all agents in the recorded driving log. We assume that the ego-vehicle is non-reactive and simply replays its driving log. Intuitively, our optimization finds controls parameters that best tracks ground-truth agent positions. Importantly, Eq 1 could be optimized over individual agents, over individual logs, or over all logs from a particular city. Motivated by our observation on city-specific behaviors, we first present results for optimizing target IDM parameters for all logs from a particular city (cf. Table 1). We highlight salient observations below. First, $\boldsymbol{\theta}^{BOS}$ and θ^{LAS} tailgate vehicles more aggressively than $\boldsymbol{\theta}^{PIT}$ and $\boldsymbol{\theta}^{SIN}$, which better tracks the city-specific min-gap statistics from Figure 1. Second, simulated agents using the default IDM world model ($\boldsymbol{\theta}^{IDM}$, optimized to minimize Eq. 1 over the entire dataset) drive less aggressively with lower *maximum acceleration* and *maximum deceleration* than city-specific models from PIT ($\boldsymbol{\theta}^{PIT}$) and BOS ($\boldsymbol{\theta}^{BOS}$). Third, $\boldsymbol{\theta}^{BOS}$ seems to have the most aggressive drivers and $\boldsymbol{\theta}^{LAS}$ seems to have least aggressive ones, as defined by *maximum acceleration* and *maximum deceleration*.

Training Log-BehaviorNet. Although each city has distinct driving characteristics, agents can still behave differently within a city. For example, Boston drivers may tailgate within the city but drive more cautiously on highways. To model this, we simply optimize Eq.1 over each individual training log. Figure 3 (a) visualizes the set of log-specific IDM parameters $\{\boldsymbol{\theta}\}$ with tSNE, color-coded by city. Rather than training BehaviorNet to directly regress these parameters, we reframe the problem as a simpler discrete classification task. Specifically, we cluster the set of $\{\boldsymbol{\theta}\}$ into K clusters and train BehaviorNet with a K-way softmax loss. We refer to this network as Log-BehaviorNet, in contrast to City-BehaviorNet. Figure 3-(b) compares the learned behavior clusters with the original city "clusters" in (a). Figure 3-(c) plots the min-gap distribution from two distinct clusters, suggesting that each cluster loosely corresponds to prototypical behaviors such as "aggressive" or "passive". We tune the number of behavior clusters (K) so as to maximize R-CLS performance on the nuPlan train-set. Interestingly, the optimal number (16) is far more than the number of distinct cities (4). Importantly, we show that city-agnostic clusters generalize better than city-specific models, particularly when evaluated on never-before-seen cities.

Incorporating Learned Priors into Rule-Based Planners. Although rule-based planners like PDM-C still outperform learning-based methods on real data, they fail to accurately model future agent behaviors in world model rollouts. We aim to bridge the gap between rule-based and learning-based planners with AdaptiveDriver (cf. Fig. 4), a model-predictive control (MPC) planner that uses behavior parameter predictions to improve the quality of world model rollouts. Notably, although AdaptiveDriver and PDM-C are both instantations of MPC-based planners, our model (1) uses a reactive world-model that (2) adapts to each log using features derived from past agent behaviors.

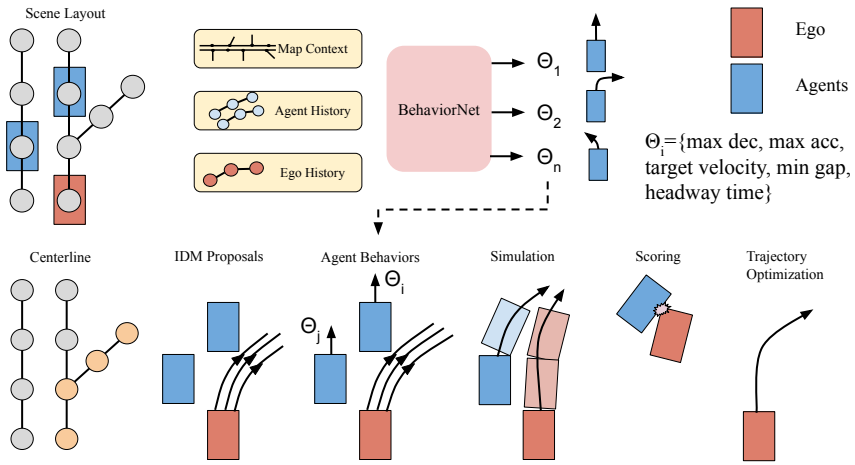

Figure 4: **Overview of AdaptiveDriver's Architecture.** AdpativeDriver's architecture extends PDM-C by replacing "world-on-rails' rollouts with adaptive reactive world models using BehaviorNet. We predict future agent behaviors parameterized as IDM controls using scene context including the ego-vehicle's history, past agent trajectories, and surrounding lane graph. Similar to PDM-C, our planner identifies the nearest centerline to the goal using graph-based search. We generate many trajectory proposals and score each according to BehaviorNet's reactive world model. The proposal with the highest score is selected and executed.

4 Experiments

In this section, we compare AdaptiveDriver with several baselines on the nuPlan benchmark and show that our proposed approach achieves state-of-the-art closed-loop planning performance. Further, we demonstrate that AdaptiveDriver generalizes to never-before-seen cities.

nuPlan Dataset. We evaluate all methods on the nuPlan benchmark [3], which includes ∼10M driving logs, collected in Boston, Pittsburgh, Las Vegas and Singapore. nuPlan identifies 73 types of interesting scenarios e.g. {`changing lanes, starting left turn, unprotected right turn, etc.`}, but only evaluates 14 scenarios in the official benchmark. We evaluate on the official Val-14 and Test-14 Hard [35] splits in Table 2, and evaluate all ablations on a representative mini

	Val-14		Test-14 Hard	
Model	NR-CLS	R-CLS	NR-CLS	R-CLS
Log Replay	94.03	75.86	85.96	68.80
IDM [5]	70.39	72.42	56.16	62.26
PDM-C [4]	92.51	91.79	65.07	75.18
RasterModel [32]	69.66	67.54	49.47	52.16
UrbanDriver [7]	63.27	61.02	51.54	49.07
GC-PGP [33]	55.99	51.39	43.22	39.63
PDM-O [4]	52.80	57.23	33.51	35.83
GameFormer [34]	80.80	79.31	66.59	68.83
PlanTF [35]	84.83	76.78	72.68	61.70
DTPP [36]	89.64	89.78	59.44	62.94
LLM-Assist$_{UNC}$ [37]	90.11	90.32	-	-
LLM-Assist$_{PAR}$ [37]	93.05	92.20	-	-
PlanAgent [38]	**93.26**	92.75	72.51	76.82
AdaptiveDriver w/ City-BehaviorNet (Ours)	92.97	93.28	70.11	78.02
AdaptiveDriver w/ Log-BehaviorNet (Ours)	93.15	**93.49**	**73.25**	**78.87**

Table 2: **nuPlan Benchmark Results**. We evaluate several rule-based and learning-based planners on the nuPlan test set. First, we note that rule-based planners like IDM and PDM-C outperform prior learning-based methods like Raster Model, LaneGCN, and UrbanDriver on NR-CLS and R-CLS. In contrast, AdaptiveDriver demonstrates that carefully combining learning-based behavior priors with rule-based planners significantly improves over prior work by 2% on Test14-Hard R-CLS. Results of prior work are reproduced from [35]. We provide additional visuals comparing our planner to PDM-C in Fig 5.

Model	City	NR-CLS	R-CLS
PDM-C [4]	PIT	92.94	90.59
	BOS	95.56	94.45
	SIN	93.49	94.15
	LAS	95.95	95.21
AdaptiveDriver w/ City-BehaviorNet (Ours)	PIT	91.48	91.94
	BOS	96.32	96.21
	SIN	95.31	95.29
	LAS	96.75	97.08
AdaptiveDriver w/ Log-BehaviorNet (Ours)	PIT	92.39	92.42
	BOS	96.39	96.44
	SIN	95.22	95.30
	LAS	96.61	97.25

Table 3: **Per-City Breakdown Analysis**. We observe that all planners with reactive world models (e.g. AdaptiveDriver w/ City-BehaviorNet and AdaptiveDriver w/ Log-BehaviorNet) consistently outperform PDM-C in closed-loop (e.g. reactive) planning performance (R-CLS) on the mini val-set. However, reactive world models do not consistently improve open-loop (e.g. non-reactive) planning performance (NR-CLS).

Model	City	NR-CLS	R-CLS
AdaptiveDriver w/ City-Oracle	All	95.10	95.20
AdaptiveDriver w/ City-BehaviorNet	All	94.97 (-0.13)	95.13 (-0.07)
AdaptiveDriver w/ Log-Oracle	All	95.29	95.47
AdaptiveDriver w/ Log-BehaviorNet	All	95.15 (-0.14)	95.35 (-0.12)

Table 4: **Oracle Target IDM Parameters**. We evaluate the quality of BehaviorNet's predictions compared to oracle target IDM parameters on the mini val-set. Specifically, City-Oracle and Log-Oracle always use the target IDM parameters (rather than predicted IDM parameters) that minimizes Eq. 1 for each log in world model rollouts. Notably, BehaviorNet closely matches the performance of the oracle on the test set, suggesting that our model accurately estimates IDM control parameters from scene context.

val-set. We construct the mini val-set by sampling 10 scenarios from all 14 scenario types across all 4 cities (560 logs).

Evaluation Setup and Metrics. The nuPlan benchmark evaluates planners under three evaluation settings: OLS, NR-CLS and R-CLS. Following prior work [35, 39, 38], we focus on NR-CLS and R-CLS as these metrics directly evaluate planning-centric performance. Specifically, *NR-CLS* and *R-CLS* measure *ego progress along expert trajectory*, *speed limit compliance*, *driving direction compliance*, *time to collision within bounds*, and *ego is comfortable*. The overall score is computed using a weighted sum of the above metrics. We refer the reader to [3] for further details.

Comparison to State-of-the-Art. We evaluate several rule-based and learning-based planners on the nuPlan test set. First, we find that AdaptiveDriver w/ Log-BehaviorNet showcases 1.7% and 3.7% improvement on R-CLS over the PDM-C baseline on Val-14 and Test-14 Hard respectively. Notably, our approach outperforms more complex LLM-based planers [37], and convincingly outperforms our PDM-C baseline on Test-14 Hard. We present a per-city breakdown analysis in Table 3. Interestingly, we find that all planners with reactive world models consistently outperform the baseline in closed-loop planning performance (R-CLS), with improvements in performance of up to 1.74%. Further, we confirm the (obvious fact) that motion planning for AVs is more difficult in some cities than in others. For example, although we achieve more than 97% closed-loop planning accuracy in LAS, we only achieve 92% performance in PIT.

Oracle Target Parameters vs. BehaviorNet Predicted Parameters. We assess the quality of BehaviorNet's predictions by comparing them to target IDM parameters from Eq. 1. The planning performance of AdaptiveDriver w/ Log-BehaviorNet closely approximates that of BehaviorNet w/ Log-Oracle, with differences of 0.14 and 0.12 in NR-CLS and R-CLS, respectively. The minimal difference between AdaptiveDriver using City-Oracle vs. City-BehaviorNet, and Log-Oracle vs. Log-BehaviorNet in Table 4, suggests that our approach can reliably predict future agent actions.

Train Cities	Test City	NR-CLS	R-CLS
PIT, BOS, SIN	LAS	96.55 (-0.06)	97.21 (-0.04)
All	LAS	96.61	97.25
PIT, BOS, LAS	SIN	95.05 (-0.17)	95.12 (-0.08)
All	SIN	95.22	95.20
BOS, SIN, LAS	PIT	91.60 (-0.79)	92.02 (-0.40)
All	PIT	92.39	92.42
PIT, SIN, LAS	BOS	96.22 (-0.17)	96.23 (-0.21)
All	BOS	96.39	96.44

Table 5: **AdaptiveDriver Generalizes to New Cities**. Although learning-based planners can leverage data-driven priors, such methods may fail when evaluated in out-of-distribution scenarios. We evaluate AdaptiveDriver's ability to generalize to previously unseen cities in the mini val-set by holding out city-specific sequences when training BehaviorNet. We find that the planning performances (NR-CLS and R-CLS) are competitive with planners trained on all cities, indicating AdaptiveDriver's generalizability.

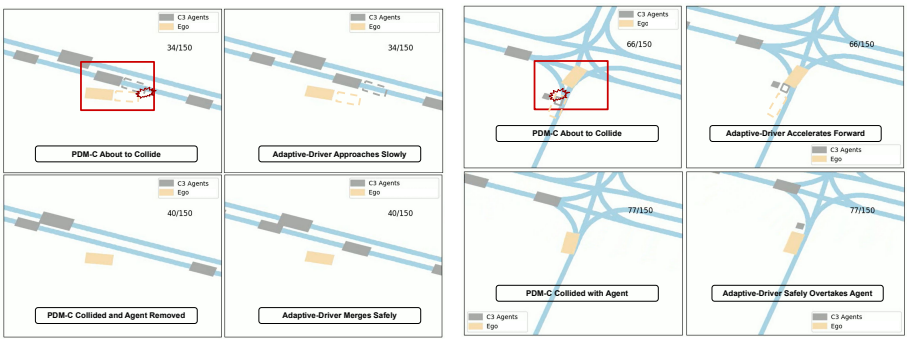

Figure 5: **Qualitative Comparison of AdaptiveDriver vs. PDM-C**. We evaluate the performance of our AdaptiveDriver against the current state-of-the-art in the simulation of two test scenarios (left and right) from the nuPlan dataset. These scenarios are 15 seconds long and sampled at 10 Hz. Timestamps are displayed on both the top and bottom rows to illustrate collision events. The yellow box represents the ego vehicle and the gray boxes represent other agents. According to nuPlan's simulation logic, vehicles that collide with each other are removed from the simulation. On the left, the ego vehicle moves from a parked position onto the road. AdaptiveDriver initially slows down to avoid a collision before smoothly entering the road, while PDM-C collides. On the right, we observe an agent about to cross the road. PDM-C drives slowly and collides, whereas our planner safely navigates forward.

AdaptiveDriver Generalizes to New Cities. As illustrated in Figure 3 (b), we observe that many agent behaviors are shared across cities. To validate this observation, we train AdaptiveDriver w/ Log-BehaviorNet on three cities and test it on a held-out city (cf. Table 5). We find that the planning scores on NR-CLS and R-CLS for seen-vs-unseen cities are comparable, suggesting that AdaptiveDriver can generalize beyond city boundaries.

5 Conclusion

In this paper, we demonstrate that each city has its own unique driving behaviors (e.g. Boston drivers tend to tailgate more than Pittsburgh drivers) and learn to model unique driving characteristics with BehaviorNet. We propose AdaptiveDriver, a model-predictive control (MPC) that unrolls and executes behavior-specific world models conditioned on BehaviorNet's predictions and achieves state-of-the-art performance on the nuPlan closed-loop reactive benchmark.

6 Acknowledgements

We would like to thank Brian Yang for his feedback on early drafts. This research was supported by Bosch Center for AI (BCAI). We thank BCAI for their financial support and resources, which made this work possible. This work was also supported in part by the NSF GRFP (Grant No. DGE2140739).

References

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. CoRL, 2017.

[2] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL https://arxiv.org/abs/1705.05065.

[3] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[4] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta. Parting with misconceptions about learning-based vehicle motion planning. *arXiv preprint arXiv:2306.07962*, 2023.

[5] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[6] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2821–2830, 2019.

[7] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, pages 718–728. PMLR, 2022.

[8] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE international conference on robotics and automation*, pages 3310–3317. IEEE, 1994.

[9] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[10] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[11] D. González, J. Pérez, V. Milanés, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4):1135–1145, 2015.

[12] C. Zhou, B. Huang, and P. Fränti. A review of motion planning algorithms for intelligent robots. *Journal of Intelligent Manufacturing*, 33(2):387–424, 2022.

[13] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[14] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, et al. Odin: Team victortango's entry in the darpa urban challenge. *Journal of field Robotics*, 25(8):467–492, 2008.

[15] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[16] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, 25(8):425–466, 2008.

[17] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.

[18] M. Buehler, K. Iagnemma, and S. Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 3642039901.

[19] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, sep 2008. ISSN 1556-4959.

[20] 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.

[21] J. Ziegler, P. Bender, T. Dang, and C. Stiller. Trajectory planning for bertha — a local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 450–457, 2014. doi:10.1109/IVS.2014.6856581.

[22] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn. Search-based optimal motion planning for automated driving. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4523–4530. IEEE, 2018.

[23] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[24] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020.

[25] J. P. Rastelli, R. Lattarulo, and F. Nashashibi. Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 510–515. IEEE, 2014.

[26] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[27] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2):267–274, 2010.

[28] A. Pongpunwattana and R. Rysdyk. Real-time planning for multiple autonomous vehicles in dynamic uncertain environments. *Journal of Aerospace Computing, Information, and Communication*, 1(12):580–604, 2004.

[29] S. Suo, S. Regalado, S. Casas, and R. Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. *CVPR*, 2021.

[30] N. Montali, J. Lambert, P. Mougin, A. Kuefler, N. Rhinehart, M. Li, C. Gulino, T. Emrich, Z. Yang, S. Whiteson, et al. The waymo open sim agents challenge. *arXiv preprint arXiv:2305.12032*, 2023.

[31] M. Althoff, M. Koschi, and S. Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726, 2017. doi:10.1109/IVS.2017.7995802.

[32] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020.

[33] M. Hallgarten, M. Stoll, and A. Zell. From prediction to planning with goal conditioned lane graph traversals. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 951–958. IEEE, 2023.

[34] Z. Huang, H. Liu, and C. Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3903–3913, 2023.

[35] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu. Rethinking imitation-based planners for autonomous driving, 2023.

[36] Z. Huang, P. Karkus, B. Ivanovic, Y. Chen, M. Pavone, and C. Lv. Dtpp: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving. *arXiv preprint arXiv:2310.05885*, 2023.

[37] S. Sharan, F. Pittaluga, M. Chandraker, et al. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*, 2023.

[38] Y. Zheng, Z. Xing, Q. Zhang, B. Jin, P. Li, Y. Zheng, Z. Xia, K. Zhan, X. Lang, Y. Chen, et al. Planagent: A multi-modal large language agent for closed-loop vehicle motion planning. *arXiv preprint arXiv:2406.01587*, 2024.

[39] B. Yang, H. Su, N. Gkanatsios, T.-W. Ke, A. Jain, J. Schneider, and K. Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following. *arXiv preprint arXiv:2402.06559*, 2024.

[40] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[41] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun. Learning lane graph representations for motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 541–556. Springer, 2020.

[42] J. Cheng, Y. Chen, X. Mei, B. Yang, B. Li, and M. Liu. Rethinking imitation-based planner for autonomous driving. *arXiv preprint arXiv:2309.10443*, 2023.

[43] S. Sharan, F. Pittaluga, M. Chandraker, et al. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*, 2023.

[44] S. Shi, L. Jiang, D. Dai, and B. Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[45] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17863–17873, June 2023.

A Implementation Details

We build upon the codebases of PDM-C [4] and nuPlan [3] for this work. We modify PDM-C [4] by adding additional longitudinal velocities and lateral offsets to the reference path. In addition, we generate additional trajectory proposals by modulating *min gap to lead agents*, *headway time*, *maximum acceleration* and *maximum deceleration*. In total, we generate 150 proposals per timestep. We train BehaviorNet for 10 epochs using the Adam optimizer [40] with a learning rate of $5\mathrm{e}{-5}$. We select a map context of radius $R = 100$m. We present BehaviorNet's architecture in Figure 6.

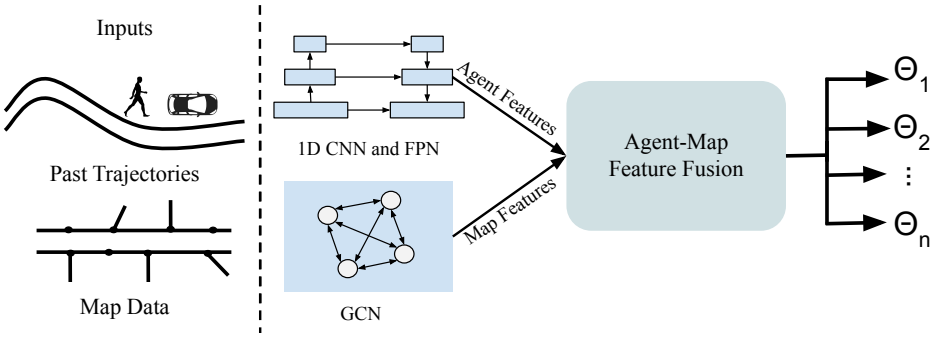

Figure 6: **BehaviorNet Architecture.** BehaviorNet uses past trajectories and map context to predict future agent behaviors parameterized as IDM controls. Following LaneGCN [41], we use a graph convolutional network (GCN) to extract map features from the lane graph. Next, we extract agent features from past trajectories. We then use LaneGCN's Agent-Map Feature Fusion to model interactions between agents and the map. Lastly, we pass these agent-map features through an MLP to predict IDM controls.

B Visualizing the Quality of Log-Specific Target IDM Parameters

We visualize the min-gap distribution of Log-BehaviorNet's rollouts in Figure 7 and find that our model more closely matches the distribution of real driving logs compared to IDM.

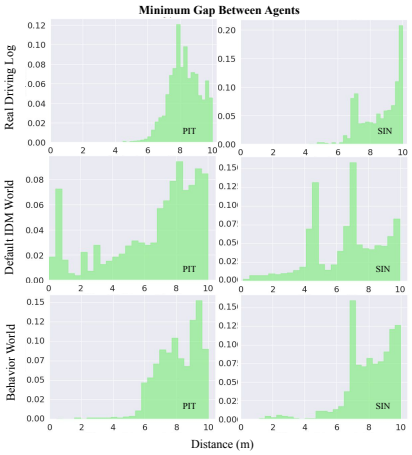

Figure 7: **Log-BehaviorNet Rollouts Closely Resemble Real-World Distributions.** We compare the gap between the ego-vehicle and lead agent (min-gap) in PIT (Pittsburgh) and SIN (Singapore) across a) real-world logs, b) Default IDM (a reactive world model optimized over the entire dataset), and c) our proposed scenario-specific world model used in *AdaptiveDriver*. We observe that our model's min-gap distribution closely aligns with the real-world distribution, outperforming prior methods, as shown in Table 2.

C AdaptiveDriver-Hybrid: Combining Open-Loop Forecasting with Closed-Loop Planning

We evaluate several rule-based and learning-based planners on the nuPlan test set in Table 2 of the main paper. We extend these results by also evaluating on the OLS metric in Table 6. First, we note that rule-based planners like IDM and PDM-C outperform prior learning-based methods like

Model	City	OLS	NR-CLS	R-CLS
Raster Model [32]	All	70.82	69.66	67.54
LaneGCN [41]	All	74.34	63.59	62.29
UrbanDriver [7]	All	80.98	63.27	61.02
IDM [5]	All	37.89	70.39	72.42
PDM-O [4]	All	82.02	52.80	57.23
PlanTF [42]	All	**89.18**	84.83	76.78
PDM-C [4]	All	66.78	92.51	91.79
LLM-Assist [43]	All	-	93.05	92.20
PlanAgent [38]	All	-	**93.26**	92.75
AdaptiveDriver w/ City-BehaviorNet (Ours)	All	64.82	92.97	93.28
AdaptiveDriver w/ Log-BehaviorNet (Ours)	All	64.03	93.15	**93.49**

Table 6: **Additional nuPlan Benchmark Results**. We extend benchmarking from Table 2 of the main paper by adding OLS metric results for completeness. We evaluate rule-based and learning-based planners on Val14 benchmark. In contrast to Table 2, where rule-based planners like IDM and PDM-C outperform prior learning-based methods like Raster Model, LaneGCN, and UrbanDriver on NR-CLS and R-CLS, learning based planners outperforms rule-based methods on OLS. Following prior work [4], we build a hybrid model that combines AdaptiveDriver with PDM-O, outperforming all prior work on NR-CLS and R-CLS, with significantly improved performance (compared to AdaptiveDriver) on OLS. We refer the reader to nuPlan [3] for metric definitions.

Raster Model, LaneGCN, and UrbanDriver on NR-CLS and C3. In contrast, learning-based planners like PDM-O and UrbanDriver outperforms rule-based planners on OLS. We observe the same trend with our proposed models: AdaptiveDriver w/ City-BehaviorNet and AdaptiveDriver w/ Log-BehaviorNet achieve OLS scores of 65.42 and 64.29 respectively, while PDM-O and UrbanDriver achieve OLS scores of 80.01 and 73.37 respectively.

Similar to PDM-H [4], we construct a hybrid model that combines AdaptiveDriver's predictions for short horizon planning (first two seconds) and PDM-O's predictions for long-term waypoints (greater than two seconds). This hybrid model (AdaptiveDriver-Hybrid w/ Log-BehaviorNet) outperforms all other methods on NR-CLS and R-CLS metrics and achieves competitive performance on OLS metric.

D BehaviorNet Regression Baseline

In the main paper, we train BehaviorNet with a K-way softmax loss to predict agent behaviors as IDM parameters from one of K behavioral clusters. However, instead of discretizing BehaviorNet's predictions into one of K classes, one can train a regressor to directly predict IDM control parameters. We ablate the training of BehaviorNet as a regressor in Table 7. Notably, the performance of AdaptiveDriver w/ Log-BehaviorNet (Regressor) is competitive with AdaptiveDriver w/ Log-BehaviorNet (Classifier). However, we posit that training BehaviorNet to directly predict control parameters may generalize better to larger numbers of behavioral clusters.

Model	City	NR-CLS	R-CLS
AdaptiveDriver w/ Log-BehaviorNet (Classifier)	All	95.15	95.35
AdaptiveDriver w/ Log-BehaviorNet (Regressor)	All	95.20	95.27

Table 7: **Regression Baseline.** We frame BehaviorNet's training objective as a K-way classification task to predict agent behavior as IDM controls in the main paper. Instead, one could train a regressor to directly predict control parameters. We note that AdaptiveDriver w/ Log-BehaviorNet (Regressor)'s performance (R-CLS of 95.27) is competitive to that of the baseline (R-CLS of 95.35) on the mini val-set.

E Limitations and Future Work.

Don't City-Specific Models Add Complexity and Limit Generalization? No! Most autonomy stacks are *already* city-specific because they make use of city-specific maps. In practice, the overhead of adding additional behavior parameters per city is minimal. More importantly, we demonstrate that AdaptiveDriver captures prototypical agent behaviors (e.g., aggressive vs. passive) that can generalize to never-before-seen cities.

What about Pedestrians and Cyclists? For simplicity, AdaptiveDriver only predicts reactive world model rollouts for cars. Pedestrians and cyclists use constant velocity rollouts (like in PDM-C). Future work should integrate reactive policies for cars, cyclists and pedestrians to better represent multi-agent interactions.

When does AdaptiveDriver Fail? Since our world models build on IDM's PID controller, we inherit its flaws. We note that IDM can be too conservative (when it mistakes a parked vehicle for a lead vehicle and stops) or too aggressive (when traveling at high speeds along a curve. Secondly, IDM only interacts with the lead vehicle, limiting multi-agent reasoning. Although we primarily focus on building city-specific and log-specific world models using the IDM, modeling agent-specific behavior models can potentially yield better performance. Future work should explore alternative optimization functions to maximize distribution similarity between real driving logs and simulated rollouts for learning control parameters. We include additional videos on our project page.

Directly Predicting Control Parameters vs. Directly Predicting Agent Behaviors. Directly predicting control parameters allows our world model to more accurately simulate multi-agent interactions in rollouts, improving proposal scoring. Further, relying on a motion controller allows us to more easily capture nuanced behaviors that may be difficult to learn like speeding up when merging in rollouts. However, relying on a rule-based controller for rollouts also implies that we inherit its limitations. Moreover, unlike learning-based approaches that directly predict agent behaviors, our approach may not significantly improve with more data. Given enough data and the right learning objectives, we posit that methods that directly predict agent behaviors may be able to mimic the characteristics of MPC-based controllers.

Rollout Accuracy vs. Runtime of BehaviorNet. BehaviorNet is limited by IDM's lateral capability, preventing it from simulating multi-agent interactions like lane changes and lane merges. In theory, one could use more sophisticated controllers that allow for lange changes and merges that differ from the reference path (like the controller in PDM-C). Alternatively, one could replace a motion controller with an ego-forecaster such as MTR++ [44] or QCNet [45]. We believe both directions are fruitful paths forward for building better world models. In practice, we find that a simple IDM-based world model already achieves state-of-the-art performance. Moreover, similar to PDM-C's constant velocity world model, our rule-based rollouts allow AdaptiveDriver to operate with lower latency. In contrast, running a SOTA trajectory prediction model at every timestep will significantly increase the latency of proposal scoring and is likely prohibitively slow for practical applications.

Learning-Based Rollouts Have Higher Latency We benchmark the latency of PDM-C and AdaptiveDriver on 1 RTX 3090 GPU with a batch size of 1 in Table E. Notably, BehaviorNet only increases AdaptiveDriver's runtime by 17ms compared to PDM-C.

Model	Latency (ms)
PDM-C	232
AdaptiveDriver w/ Log-BehaviorNet	249

Table 8: **Latency Analysis.** We compare the latency of PDM-C and AdaptiveDriver w/ Log-BehaviorNet. Unsurprisingly, we find that AdaptiveDriver is slower than PDM-C due to our learning-based rollouts.