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 $5e{-}5$. We select a map context of radius $R = 100$m. We present BehaviorNet's architecture in Figure 6.

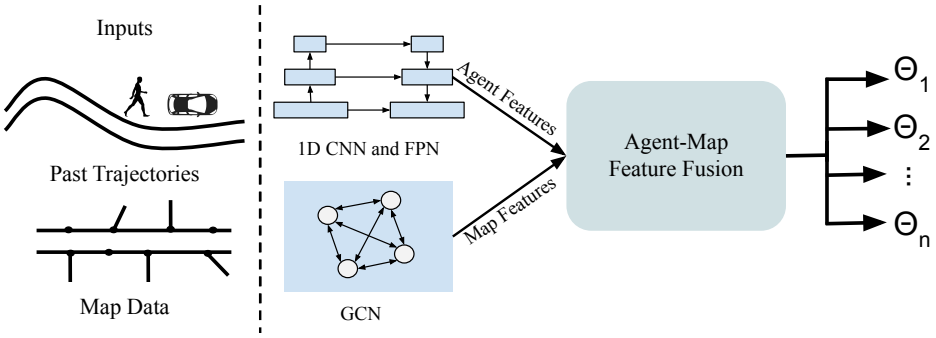

Figure 6: **BehaviorNet Architecture.** BehaviorNet uses past trajectories and map context to predict future agent behaviors parameterized as IDM controls. Following LaneGCN [41], we use a graph convolutional network (GCN) to extract map features from the lane graph. Next, we extract agent features from 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.

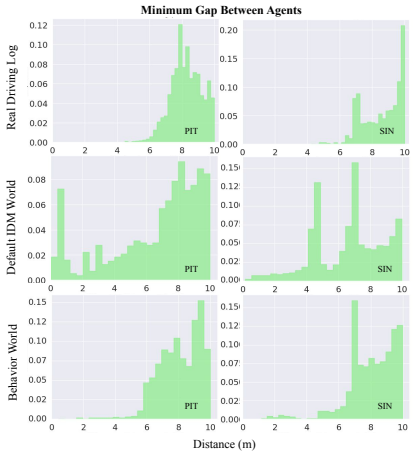

Figure 7: **Log-BehaviorNet Rollouts Closely Resemble Real-World Distributions.** We compare the gap between the ego-vehicle and lead agent (min-gap) in PIT (Pittsburgh) and SIN (Singapore) across a) real-world logs, b) Default IDM (a reactive world model optimized over the entire dataset), and c) our 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.