Hidden Biases of End-to-End Driving Datasets

Julian Zimmerlin Jens Beißwenger Bernhard Jaeger Andreas Geiger Kashyap Chitta
University of Tübingen Tübingen AI Center
zim.julian@gmail.com, jensbeiswenger@gmail.com
{bernhard.jaeger, kashyap.chitta, a.geiger}@uni-tuebingen.de

Abstract

End-to-end driving systems have made rapid progress, but have so far not been applied to the challenging new CARLA Leaderboard 2.0. Furthermore, while there is a large body of literature on end-to-end architectures and training strategies, the impact of the training dataset is often overlooked. In this work, we make a first attempt at end-to-end driving for Leaderboard 2.0. Instead of investigating architectures, we systematically analyze the training dataset, leading to new insights: (1) Expert style significantly affects downstream policy performance. (2) On complex driving datasets, frames should not be weighted based on simplistic criteria such as class frequencies. (3) Instead, estimating whether a frame changes the target labels compared to previous frames can reduce dataset size without losing important information. By incorporating these findings, our model ranks first and second respectively on the map and sensors tracks of the 2024 CARLA Challenge.

1. Introduction

Imitation Learning (IL) for end-to-end autonomous driving has seen great success in recent work on the CARLA simulator [4], particularly in the Leaderboard 1.0 setting. A key ingredient contributing to this success is the inherent scalability of IL with increased training data, which is now straightforward to collect on Leaderboard 1.0 as a result of steady progress in planning algorithms for CARLA [2, 3, 6–8, 11, 13, 15, 17, 19]. However, with the introduction of Leaderboard 2.0, driving models now face 38 new complex scenarios. These often require driving at high speeds, deviating off the center of the lane, or handling unexpected dynamic obstacles. The best planning algorithm from Leaderboard 1.0 [8] fails to solve these new scenarios, making it significantly harder to collect high-quality driving demonstrations needed for training IL models. As a result, there are no existing IL based methods for Leaderboard 2.0.

In this work, we present the first attempt at tackling Leaderboard 2.0 with end-to-end IL. We leverage the recently open-sourced PDM-Lite [1] planner, which can solve the new Leaderboard 2.0 scenarios, to automatically collect high quality datasets. To gain insights about the challenges posed by this new task, we use a simple existing IL model, TransFuser++ [8], with minimal changes to its architecture and training objective. Instead, we focus on a critical but understudied aspect of IL – *the training dataset*. In particular, the impact of factors besides dataset scale, such as the diversity of the training distribution, is nuanced and not yet well understood. We conduct a systematic analysis of our driving dataset, leading to multiple new insights.

First, the expert's *driving style*, in addition to its performance, significantly influences its suitability for IL. To develop an effective expert, it is important to base the expert's behavior on signals that are easily observable and interpretable by the IL policy, rather than relying excessively on privileged inputs. This behavior also resembles how human drivers perceive and react to their environment.

Second, we find the use of frequency-based *class weights*, a common approach to facilitate learning of classification tasks on imbalanced datasets, detrimental for target speed prediction in autonomous driving. Over-represented classes do not represent a single "uninteresting" mode of the data distribution– in contrast, they may contain a mixture of both uninteresting (e.g., braking while waiting at red lights) and crucial parts of the dataset (e.g., braking for obstacles).

Finally, we study *data filtering* as an alternative means to assigning the importance of frames, by which we reduce our dataset size by $\sim 50\%$ while maintaining performance.

Based on these findings, we train a model which safely handles urban driving in diverse scenarios and ranks first on the map track and second on the sensors track of the 2024 CARLA challenge. Our dataset, code, and pre-trained models will be made available as a starter kit for the community.

2. Preliminaries

We consider the task of urban navigation along routes with complex scenarios. Each route is a list of GNSS coordinates called target points (TPs) which can be up to 200 m apart.

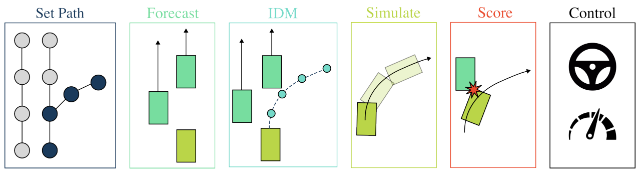

Figure 1. **PDM-Lite** [1]. This open-source rule-based planner solves all 38 scenarios of CARLA Leaderboard 2.0.

Metrics. We use the CARLA online metrics. Our main metric is the Driving Score (DS) which multiplies Route Completion (RC) with the Infraction Score (IS). RC is the percentage of the route completed. IS is a penalty factor at 1.0 that gets reduced multiplicatively for every infraction.

Benchmark. To evaluate agents, Leaderboard 2.0 provides 20 *official validation* routes on Town13 which on average are 12.39 km long and contain 93 scenarios. We split them into short routes, each containing only a single scenario. This allows for more accurate performance evaluation per scenario type. After splitting, we sample up to 15 routes per scenario type without replacement to create the *Town13 short* benchmark. There are 38 scenario types, but in some cases, fewer (or no) routes are available, which gives a total of 400 routes from 36 scenarios in this benchmark. As the calculation of *MinSpeedInfractions* is unsuited to short routes, we exclude them from the IS metric on Town13 short. We reproduce TransFuser++ [8] on this benchmark using data collected with the PDM-Lite expert [1], which are summarized in the following. We choose PDM-Lite as it achieves state-of-the-art DS on the official validation routes. Unlike other concurrent Leaderboard 2.0 planners [9, 18], it is also publicly available and possible to modify.

PDM-Lite [1] is a privileged rule-based approach for collecting data in Leaderboard 2.0 capable of tackling all 38 new scenarios. It consists of six stages (Fig. 1).

- First, it creates a dense path of spatially equidistant points using the A* planning algorithm, given sparse TPs from the simulator. For new scenarios that require leaving this path, a short segment of the route where the scenario will be spawned is shifted laterally towards an adjacent lane.
- It forecasts dynamic agents for 2s into the future, assuming that they maintain their previous controls.
- It selects a leading actor and generates a target speed proposal using the Intelligent Driver Model [16].
- The target speed proposal is converted into an actual expected sequence of ego-vehicle bounding boxes in closed-loop by using a kinematic bicycle model.
- Having forecasted all actors, it checks for bounding box intersections between the simulated ego vehicle and other vehicles. It scores the ego vehicle's motion accordingly: if it detects an intersection, it rejects the IDM target speed proposal, and sets the target speed to zero.
- The steering value is estimated with a lateral PID controller, which minimizes the angle to a selected point along the path ahead. For the throttle and brake predictions, it employs a linear regression model using features extracted based on the current speed and target speed.

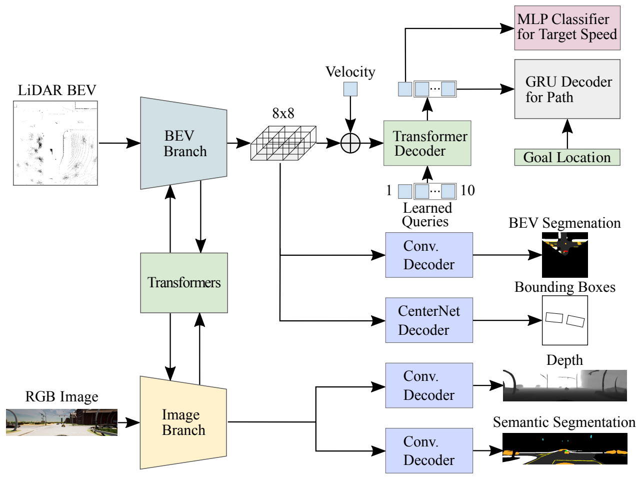

Figure 2. **TF++** [8]. This end-to-end imitation learning approach is the best publicly available baseline for CARLA.

TransFuser++ [8] is the best-performing open-source model on Leaderboard 1.0 (Fig. 2). Given sensor inputs, it predicts a target speed and desired path which are input to a controller module to drive the vehicle. We require two changes compared to [8] for compatibility with PDM-Lite:

- **Two-hot labels.** While the rule-based planner in [8] uses only 4 different target speed classes up to 8m/s (28.8km/h), PDM-Lite operates with a continuous range of target speed values up to 20m/s (72km/h). To solve target speed regression with a classification module, we employ *two-hot labels* [5]. This method converts a continuous value into a two-hot representation by interpolating between one-hot labels of the two nearest classes. For instance, with our 8 speed classes ([0.0, 4.0, 8.0, 10, 13.89, 16, 17.78, 20] m/s), a target speed of 3.0m/s is represented as $[0.25, 0.75, 0, 0, 0, 0, 0, 0]$. These speed classes were selected by analyzing the distribution of target speeds chosen by PDM-Lite in our dataset.
- **Dynamic lookahead controller.** For stable lateral control at the high speeds required by Leaderboard 2.0, it is advantageous to adjust the distance of the point selected to follow along the ego vehicle's predicted path based on the current speed. TF++ predicts a set of 10 checkpoints, each spaced 1m apart, with the first checkpoint located 2.5 meters from the vehicle center. The distance of the checkpoint to which the lateral controller minimizes the angle is determined by the formula $d = (0.097v+0.692)$, where v is the ego's speed in km/h. We round down to the nearest available predicted checkpoint. This scaling ensures that at low speeds, the controller selects a closer point, facilitating tight turns, while at high speeds, it selects a distant point, resulting in more stable steering.

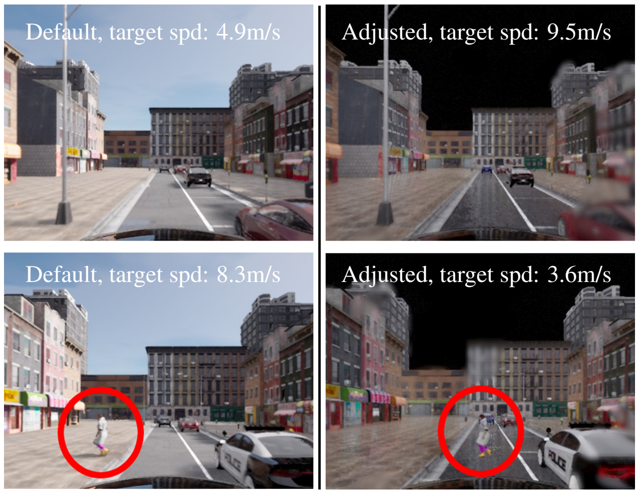
Default, target spd: 4.9m/s
Adjusted, target spd: 9.5m/s
Default, target spd: 8.3m/s
Adjusted, target spd: 3.6m/s

Figure 3. **Expert style compared on the same route.** The default PDM-Lite brakes early (left), when the pedestrian is hardly visible in the image, while the adjusted expert brakes later (right).

3. Hidden Biases

Expert style. While expert *performance* is often reported in prior work, the manner in which it achieves that performance, i.e., expert *style*, is often overlooked. Although harder to quantify, it is an important aspect to consider for IL. For instance, consider PDM-Lite's behavior when approaching pedestrians (Fig. 3). By default, it addresses this scenario by slowing down when the pedestrian is predicted to enter the driving path. However, at this point, the pedestrian is usually still obstructed by a parked vehicle. With our adjusted IDM parameters, the expert brakes rather sharply instead, coming to a stop at a distance of roughly 4 meters in front of the more visible pedestrian. This leads to a $\sim 4\times$ decrease in pedestrian collisions for models trained on the adjusted expert data. Notably, this update does not affect the expert's own pedestrian collision rate. The improvement is likely due to the adjusted behavior providing a clear braking signal for the model to learn from (a pedestrian directly in front of the ego vehicle), whereas the default behavior requires the model to generalize across various situations where a pedestrian *might* appear further ahead.

Target speed weights. Training TF++ involves using class weights in a target speed classification loss, which are calculated anti-proportionally to the number of occurrences of the respective class in the dataset [8]. This means that classes that appear frequently get a lower weight than those that appear rarely. We find that removing these weights significantly improves the performance of TF++ on our task (Table 1). We believe this is due to the weight of class 0 (braking), the most common in the dataset. While some part of the data for class 0 is redundant (e.g., waiting at red lights), some frames are among the most crucial for avoiding infractions, such as coming to a stop in front of stop signs or pedestrians. With a low weight on the target speed loss for these frames, ignoring short braking phases in these situations is an easy shortcut for the model to fall into.

Speed Weights	DS↑	RC ↑	IS ↑
✓	82 ± 1	98 ± 0	0.83 ± 0.01
✗	85 ± 0	99 ± 0	0.85 ± 0.00

Table 1. **Speed weights.** Results on Town13 short, reported over 3 training seeds. Weights: [0.29, 1.30, 0.69, 0.81, 4.43, 4.76, 3.90, 2.41] for the speeds [0.0, 4.0, 8.0, 10, 13.89, 16, 17.78, 20] m/s.

Data filtering. As an alternative to measure importance of frames, we propose the use of heuristics that estimate whether a frame changes the model's target labels compared to previous frames. More precisely, we keep all frames that change the target speed by more than 0.1m/s, or the angle to any of the predicted path checkpoints by more than 0.5° compared to the previous frame ($\tilde{4}0\%$ of all frames). Additionally, we randomly retain 14% of the remaining frames and discard the rest, for a total filtered dataset containing 51% of all available frames. We then train with $2\times$ the number of epochs to keep the total number of gradient updates similar. In Table 2, we present the results of our proposed filtering strategy on the official Leaderboard. Our result on the sensors track (5.2 DS) is from a model without filtering, while the result on the map track (5.6 DS) is with filtering. By reducing the dataset size by 49%, with slightly improved performance, we show that our heuristic effectively removes redundant frames without losing information.

Model	DS ↑	RC ↑	IS ↑
LRM [14]	1.2	9.6	0.31
Kyber-E2E [18]	3.5	8.5	0.50
CarLLaVA	6.9	18.1	0.42
TF++ (no filtering)	5.2	11.3	0.48
TF++ (w/ filtering)	5.6	11.8	0.47

Table 2. **Filtering improves scores on CARLA Leaderboard 2.0.** Secret test routes (Town 14). TF++ (Ours) outperforms prior modular pipelines [14, 18], and places 2nd overall.

4. Additional Experimental Details

Training dataset. The CARLA team provides 90 training routes on Town12 with a total length of 780.6km. We split the training routes into smaller routes containing one scenario each. The scenario distribution is shown in Figure 4. We sample from these routes with replacement to obtain a set of 50 routes per scenario, on which we collect a training dataset using our expert driver (198k frames). The dataset contains RGB images with a resolution of 384x1024 pixels, LiDAR point clouds, and the training labels needed

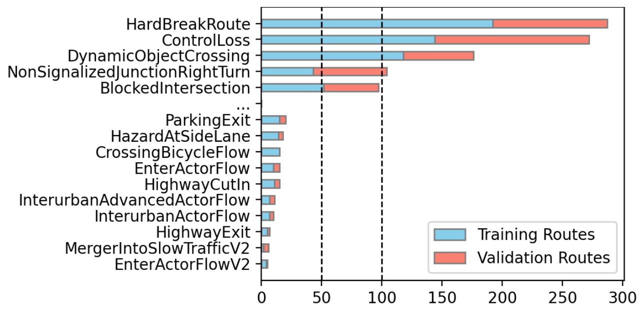

Figure 4. **Scenario distribution** in the available long routes.

for TF++ (path checkpoints, expert target speed, and auxiliary labels such as BEV semantics and vehicle/pedestrian bounding box predictions). Additionally, we collect data on Towns 01-05 and 10, which contain the six scenarios from Leaderboard 1.0 (139k frames), for a total of 337k frames. For the models in Table 2, we also include training data from the provided validation routes on Town13 (20 routes, total length 247.6km, again upsampled to 50 routes per scenario), adding 194k frames (531k in total).

Implementation. We use a cosine annealing learning rate schedule [10] with $lr_0 = 3 \cdot 10^{-4}, T_0 = 1, T_{mult} = 2$ and train our models for 31 epochs. We train each model on four A100 GPUs with a total batch size of 64, which takes 2-3 days depending on the architecture. Models marked with:

- "*Big*" use the default regnety_032 [12] architecture of TF++ for the image and LiDAR perception modules instead of ResNet34 used in our "Base" setting.
- "*Pre*" use two-stage training, where we first pre-train exclusively with perception losses (BEV semantics, bounding boxes, image depth, image semantics) for 15 epochs, before training for 31 epochs with all losses.
- "*Ens*" are ensemble models, averaging predictions from 3 models trained with different random seeds.

Ablations. In Table 3, we present a number of ablations and additional expriments for the "Base" model and compare to expert performance. Ensembling ("Ens") and two-stage training ("Pre") provide small improvements. To react better to vehicles that are further away, we extend the LiDAR range in front of the ego to 64m from 32m ("L64m"). This also resulted in a small improvement, but we do not use it in our leaderboard submissions. Giving the next two target points as input instead of only one ("2TPs") fails to increase performance. Our final models (used in Table 2 and Table 4) combine the benefits of "Big", "Pre", and "Ens".

Early termination. It can be beneficial to stop an agent preemptively, reducing RC and increasing IS to improve DS. We formulate the expected DS of an agent as $100xI^{xL}$ where $x \in [0, 1]$ is the fraction of the route that the agent completes, L is the route length, and $I = 0.5^{CP} * 0.6^{CV} * 0.65^{CL} * 0.7^{RL} * 0.8^{SI} * 0.7^{ST} * 0.7^{YE}$ is the expected infraction coefficient per km, including all non-negligible infraction types. Maximizing this function, we obtain the solution $x = -(L \log I)^{-1}$. A model profits from early termination if $x < 1$, i.e. $I < 0.907$. With $L = 10.295$ (mean length of test routes) and $I = 0.43$ (estimated on validation routes), we obtain $x = 0.115$. Thus, our model should theoretically stop at $d = xL = 1.18\text{km}$ to maximize expected DS. Since I and L are only estimates, we set target speed to 0 after $d = 1.5\text{km}$ in practice. We track distance traveled using the agent's speed sensor. Early termination has a significant effect on the scores obtained on long Town13 validation routes, as shown in Table 4. As Table 2 shows, all good submissions to the Leaderboard 2.0 test server have less than 18.1 RC, which implies that all these methods use a variant of early termination either explicitly or implicitly.

Setting	DS ↑	RC ↑	IS ↑
Base	85 ± 0	99 ± 0	0.86 ± 0.00
Big	85 ± 1	98 ± 1	0.86 ± 0.01
Pre	87 ± 1	98 ± 0	0.87 ± 0.01
Ens	86	98	0.87
L64m	86	97	0.87
2TPs	82	96	0.85
Expert	*99*	*100*	*0.99*

Table 3. **Results on Town13 short.** Std over 3 training seeds where available. Training on Towns 01-05, 10, 12 (337k frames).

Early termination	DS↑	RC ↑	IS ↑
✗	0.70	70.1	0.04
✓	4.81	8.0	0.59

Table 4. **Early termination** after 1km on the 20 validation routes.

5. Discussion

In repeated Leaderboard submissions, we observed significant variance in DS, with identical submissions yielding results that differ by more than 1DS. In addition, as shown in Table 4, DS is influenced significantly by early termination on long routes, which does not reflect any actual improvement in driving behavior. Our formula offers a way to compute an appropriate driving distance for the test server, however, it is necessary to consider results from other benchmarks before drawing conclusions based on DS.

Conclusion. With a systematic analysis of training dataset biases for end-to-end driving in CARLA, we reveal the significant impact of expert style on IL policy performance. We further provide insights into the challenges of assigning importance to frames through weighting or filtering, and provide a simple and effective heuristic that estimates importance based on changes in target labels. We reproduce TransFuser++ in the Leaderboard 2.0 setting, providing the first recipe for training an end-to-end driving system that attains non-trivial performance. We hope this can serve as a starting point for future research on this benchmark.

Acknowledgements. Bernhard Jaeger and Andreas Geiger were supported by the ERC Starting Grant LEGO-3D (850533) and the DFG EXC number 2064/1 - project number 390727645. Kashyap Chitta was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Bernhard Jaeger and Kashyap Chitta, and Katrin Renz for the useful exchange.

References

[1] Jens Beißwenger. Pdm-lite: A rule-based planner for carla leaderboard 2.0. Technical report, University of Tübingen, 2024. https://github.com/autonomousvision/carla_garage/blob/leaderboard_2/doc/report.pdf. 1, 2

[2] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *CoRL*, 2019. 1

[3] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *PAMI*, 2023. 1

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017. 1

[5] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024. 2

[6] Shu Ishida, Gianluca Corrado, George Fedoseev, Hudson Yeo, Lloyd Russell, Jamie Shotton, João F. Henriques, and Anthony Hu. Langprop: A code optimization framework using language models applied to driving. *arXiv.org*, 2401.10314, 2024. 1

[7] Bernhard Jaeger. Expert drivers for autonomous driving. Master's thesis, University of Tübingen, 2021.

[8] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *ICCV*, 2023. 1, 2, 3

[9] Qifeng Li, Xiaosong Jia, Shaobo Wang, and Junchi Yan. Think2drive: Efficient reinforcement learning by thinking in latent world model for quasi-realistic autonomous driving (in carla-v2). *arXiv.org*, 2402.16720, 2024. 2

[10] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 4

[11] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multimodal fusion transformer for end-to-end autonomous driving. In *CVPR*, 2021. 1

[12] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *arXiv:2003.13678*, 2020. 4

[13] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, Almut Sophia Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level representations. In *CoRL*, 2022. 1

[14] Luis Alberto Rosero, Iago Pachêco Gomes, Júnior Anderson Rodrigues da Silva, Carlos Andre Braile Przewodowski Filho, Denis Fernando Wolf, and Fernando Santos Osório. Integrating modular pipelines with end-to-end learning: A hybrid approach for robust and reliable autonomous driving systems. *Sensors*, 2024. 3

[15] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *CoRL*, 2022. 1

[16] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 2000. 2

[17] Peng Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*, 2022. 1

[18] Weize Zhang, Mohammed Elmahgiubi, Kasra Rezaee, Behzad Khamidehi, Hamidreza Mirkhani, Fazel Arasteh, Chunlin Li, Muhammad Ahsan Kaleem, Eduardo R. Corral-Soto, Dhruv Sharma, and Tongtong Cao. Analysis of a modular autonomous driving architecture: The top submission to carla leaderboard 2.0 challenge. *arXiv.org*, 2405.01394, 2024. 2, 3

[19] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *ICCV*, 2021. 1