Hidden Biases of End-to-End Driving Models

Bernhard Jaeger Kashyap Chitta Andreas Geiger
University of Tübingen Tübingen AI Center
{bernhard.jaeger, kashyap.chitta, a.geiger}@uni-tuebingen.de

Abstract

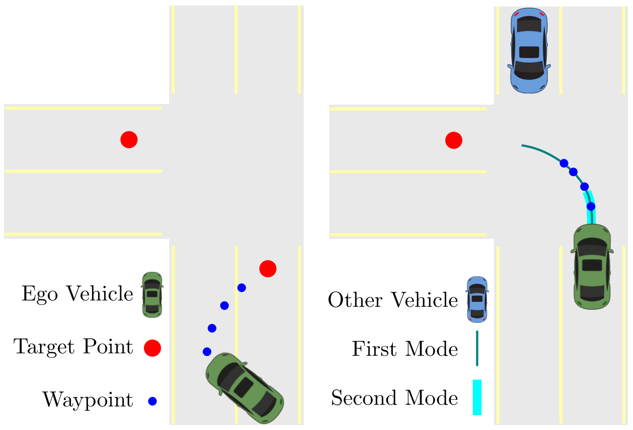
End-to-end driving systems have recently made rapid progress, in particular on CARLA. Independent of their major contribution, they introduce changes to minor system components. Consequently, the source of improvements is unclear. We identify two biases that recur in nearly all state-of-the-art methods and are critical for the observed progress on CARLA: (1) lateral recovery via a strong inductive bias towards target point following, and (2) longitudinal averaging of multimodal waypoint predictions for slowing down. We investigate the drawbacks of these biases and identify principled alternatives. By incorporating our insights, we develop TF++, a simple end-to-end method that ranks first on the Longest6 and LAV benchmarks, gaining 11 driving score over the best prior work on Longest6.

1. Introduction

End-to-end driving approaches have rapidly improved in performance on the CARLA leaderboard [1], the de-facto standard for fair online evaluation. Driving scores have increased from under 20 [9, 22] to over 70 [26, 31] in just two years. However, why recent systems work so well is not fully understood, as both methods and training sets differ largely between submissions. Rigorous ablations are expensive due to the large design space of driving systems and the need to simulate large amounts of driving for evaluation.

In particular, recent methods trained with Imitation Learning (IL) have shown strong performance [6, 8, 11, 26, 31]. They are trained using offline datasets, yet they can surprisingly recover from the classic compounding error problem of IL [21, 24], as indicated by their high route completions [8, 11, 26, 31]. While they do not utilize HD maps as input, they are provided with map-based GNSS locations in the center of the lane (spaced 30 m apart on average) called target points (TPs) that describe the route the car should follow. TPs were introduced as an alternative form of conditioning signals to convey driver intent [9, 22]. Prior work [13, 14] used discrete navigation commands or NCs (i.e. follow lane, turn right, ...) instead.

In this paper, we show that TP conditioned models re-



(a) Target point shortcut (b) Waypoint ambiguity

Figure 1: **Hidden biases.** (a) When outside their training distribution, current methods extrapolate waypoint predictions to the nearest target point, helping them recover. (b) The future velocity is multi-modal, but current methods commit to a single plan, which leads to interpolation.

cover from the compounding error problem because they use geometric information contained in the TP to reset steering errors periodically (at every TP). This makes them implicitly rely on accurate map information, even though they are otherwise HD map free. Steering directly towards a TP is a shortcut [16] that these IL methods learn to exploit. When methods accumulate enough steering error to be out of distribution during deployment, we observe that they steer towards the nearest TP. When the TP is close, this has the effect of driving back to the lane center, where it is in distribution again. This is illustrated in Fig. 1a. However, when the TP is far away, this shortcut can lead to catastrophic steering errors (e.g. cutting a turn). We show examples of this behavior for various SotA architectures in Section 3.1. We demonstrate that the shortcut problem is intrinsically related to the decoder architecture and that a transformer decoder [29] can mitigate it.

Another common aspect of the current SotA is that they use waypoints (future positions of an expert driver) as output representations [8, 11, 31]. We point out that this is an ambiguous representation as the future velocity is multi-modal, yet the model commits to a point estimate. This is illustrated in Fig. 1b. We show that this ambiguity can sometimes be helpful due to the continuous nature of waypoints: the network can continuously interpolate between modes. We propose an alternative that explicitly predicts the uncertainty of the network via target speed classification, and show that interpolating between target speeds weighted by the uncertainty reduces collisions. Our controlled experiments also cover important but sometimes neglected details in the training of end-to-end driving systems, such as augmentation, training schedules, and dataset size. In particular, we revisit the idea of shift and rotation augmentations to aid recovery [2,5]. These were common in early IL methods for CARLA with control outputs [14], but are harder to implement with waypoint outputs and not used by the current SotA. We find that they yield significant improvements.

Using these insights, we develop TransFuser++ (TF++), which sets a new SotA on the Longest6 [11] and LAV [8] benchmarks. Two of the ideas we apply, transformer decoder pooling and path-based outputs instead of waypoints, have been used in Interfuser [26]. However, these are presented as minor details, and their impact is not studied in isolation as in our work. TF++ is significantly simpler than Interfuser, yet outperforms it by a large margin on Longest6, as we show in Section 4. We use $\sim 4\times$ less data, 1 camera instead of 4, and do not require complex heuristics to extract throttle and brake commands for our path output.

Contributions:

- We show that target point conditioned models learn a shortcut that helps them recover from steering errors.
- We point out that the waypoint output representation is ambiguous, but its continuous nature helps models collide less by interpolating to slow down.
- Using the insights gained from controlled experiments, we propose TransFuser++ which places first on the Longest6 and LAV benchmarks.

Code, data, and models are available at https://github.com/autonomousvision/carla_garage.

2. Related Work

IL for autonomous driving dates back over 30 years [20]. It regained traction with the seminal works of [3, 5, 13] and the release of CARLA [15], a 3D simulator used in numerous recent research advances in autonomous driving [9, 10, 14, 17, 19, 25, 28, 33]. Early IL approaches evaluated in CARLA used a discrete navigation command (NC) [7, 9, 14], but their performance is not competitive to modern approaches [8, 11, 26, 31] which predict waypoints conditioned on TPs. In this work, we revisit an idea used in early systems but neglected in modern TP-conditioned methods: geometric shift augmentations [5, 9, 14, 21].

LAV [8] supervises waypoint outputs with additional data by making predictions for other nearby agents in the scene. Their waypoints are initially generated with NC conditioning. These are then refined using a GRU [12]. They observe a large (+50) improvement in route completion from this refinement. Our findings suggest that the refinement module improves steering primarily through TP conditioning, which is only provided to the refinement GRU. **TCP** [31] observes that waypoints are stronger at collision avoidance than directly predicting controls, but sometimes suffer at large turns. They leverage the strength and mitigate the weaknesses of these two representations by creating a situation dependent ensemble. Our study suggests that the waypoints are better at collision avoidance due to an implicit slowdown when the car is uncertain. **PlanT** [23] investigates planning on CARLA by processing object bounding boxes with a transformer. Its sensorimotor version differs from other SotA models in that it is trained in two stages (not end-to-end). In our setting, we find that end-to-end training is crucial. We show that their observations about dataset scale also hold for end-to-end models.

TransFuser [11, 22] is a simple, well-known and widely used baseline for CARLA. We provide an explanation for the large differences in route completion between the target point conditioned TransFuser architecture and its NC conditioned baselines [7, 9, 14]. Further, we propose modifications to its architecture, output representation and training strategy which lead to significant improvements.

Interfuser [26] regresses a path for steering, predicts object density maps, and classifies traffic rule flags. This representation is converted by a forecasting mechanism and hand designed heuristics into control. Like our proposed method, it uses a transformer decoder for pooling features, and disentangles future velocities from the path in the output. While these ideas were already present, they were not studied or discussed as significant. Our work adds to the literature by showing that these design choices are indeed critical to performance and providing explanations as to why. Furthermore, our final system is simpler and significantly outperforms Interfuser on Longest6.

ReasonNet [27], **ThinkTwice** [18] and **CaT** [32] are concurrent end-to-end driving approaches that all use TP conditioning and waypoints as output representation.

3. Hidden Biases of End-to-End Driving

We consider the task of urban navigation from point A to B [11]. The goal is to complete routes with dense traffic, multiple lanes and complex geometries (e.g. roundabouts) without incurring infractions. Along the way, the agent encounters manually designed pre-crash traffic scenarios. Each route is a list of GNSS coordinates called target points (TPs) which can be up to 50 m apart ($\sim$30m on average).

Metrics: We use the CARLA online metrics. Route Completion (RC) is the percentage of the route completed. Infraction Score (IS) is a penalty factor starting at 1.0 that

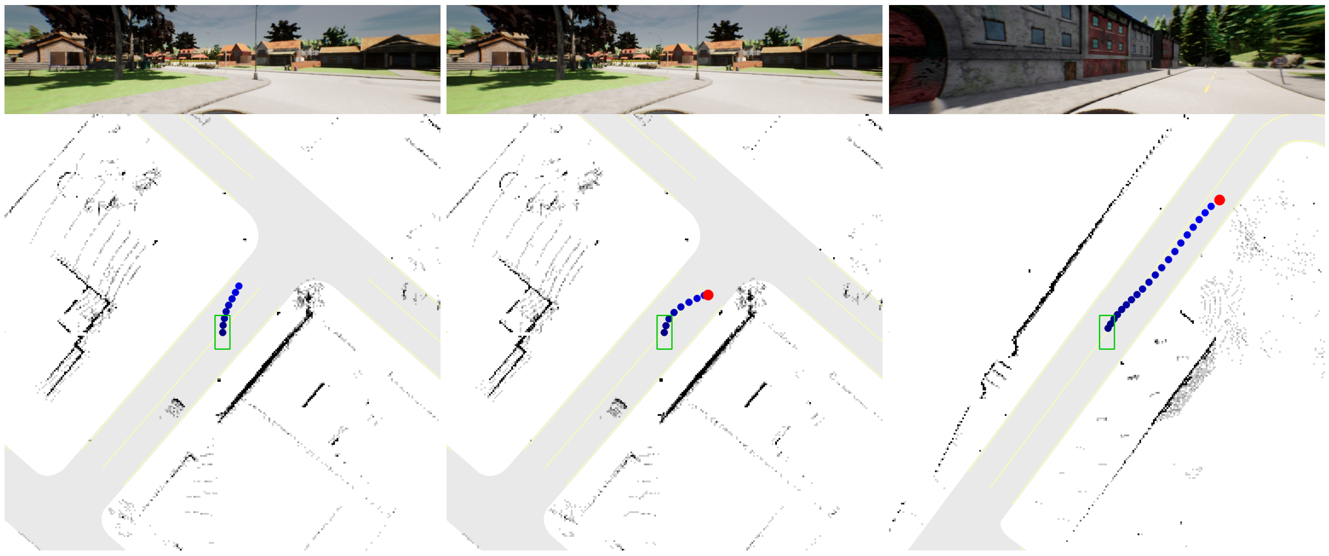

(a) TransFuser (NC conditioned) (b) TransFuser (TP conditioned) (c) LAV [8] (TP conditioned)

Figure 2: **Extrapolation to target point.** In unknown situations, TP conditioned methods extrapolate their waypoints towards target points. This periodically resets steering errors and is a form of implicit map based recovery. However, relying on extrapolation is a shortcut that can lead to catastrophic errors in certain situations (see Fig. 3a and Fig. 3b).

gets reduced multiplicatively for every infraction. Our main metric is the Driving Score (DS) which multiplies RC with the IS. Where insightful, we report infraction per kilometer metrics. For a comprehensive description, refer to [11].

Baseline: As a baseline, we reproduce TransFuser [11]. This is a simple method representative of current SotA driving systems on CARLA. The main differences in our reproduction are a 360° field of view (FOV) LiDAR to enable safe lane changes and a dataset where the expert is driving 2× faster to speed up evaluation. A list of other minor changes is provided in the supplementary material.

Benchmark: For all experiments in this section, we train on CARLA towns 01, 03, 04, 06, 07 and 10. We use the 16 validation routes from LAV [8] in the withheld towns 02 and 05 for evaluation. In order to reduce the influence of training and evaluation variance which can be large in CARLA [4, 11], every experiment reports the average of 3 training seeds, evaluated 3 times each. We additionally report the training standard deviation for the main metrics.

3.1. A shortcut for recovery

While current SotA CARLA methods have fundamentally different architectures [8,11,26,31], they are all trained with conditional IL using fixed pre-recorded datasets. The evaluation task involves challenging routes which are ~1.5km long, so it would be expected that these methods suffer from *compounding errors*, a well-known problem for IL [24]. Geometric shift augmentations [5] are a common approach to teach IL methods how to recover from such compounding steering errors [5, 14]. Surprisingly, even though such augmentations are not employed by SotA methods on CARLA, they report high RCs, demonstrating that they are not prone to this expected failure mode.

All the aforementioned methods condition their predictions using the next target point (TP) along the route. To understand the importance of the TP, we train 2 versions of our reproduction of TransFuser: one with the original TP conditioning and another with a discrete (NC) condition. This is a one hot vector that indicates whether the car should follow the lane, turn right at the next intersection, etc. It contains no geometric information about the center of the lanes, but still removes the inherent task ambiguity of inner-city driving [13]. For the NC conditioned model, we also implement shift and rotation augmentations by collecting augmented frames during data collection. We deploy a second camera in the simulator, that changes its position and orientation randomly at every time-step. The output labels are transformed accordingly for training the model (implementation details and examples can be found in the supplementary material). The results are shown in Table 1.

Cond.	**Aug.**	**DS** ↑	**RC** ↑	**Dev** ↓
NC	-	32 ± 8	56 ± 12	0.86
NC	✓	35 ± 3	54 ± 4	0.99
TP	-	**39** ± 9	**84** ± 7	**0.00**

Table 1: **Conditioning and Augmentation.**

We observe a significant difference in RC of 28 points when switching between TP and NC conditioning. The TP conditioned model has 0 route deviations per km ("Dev"),

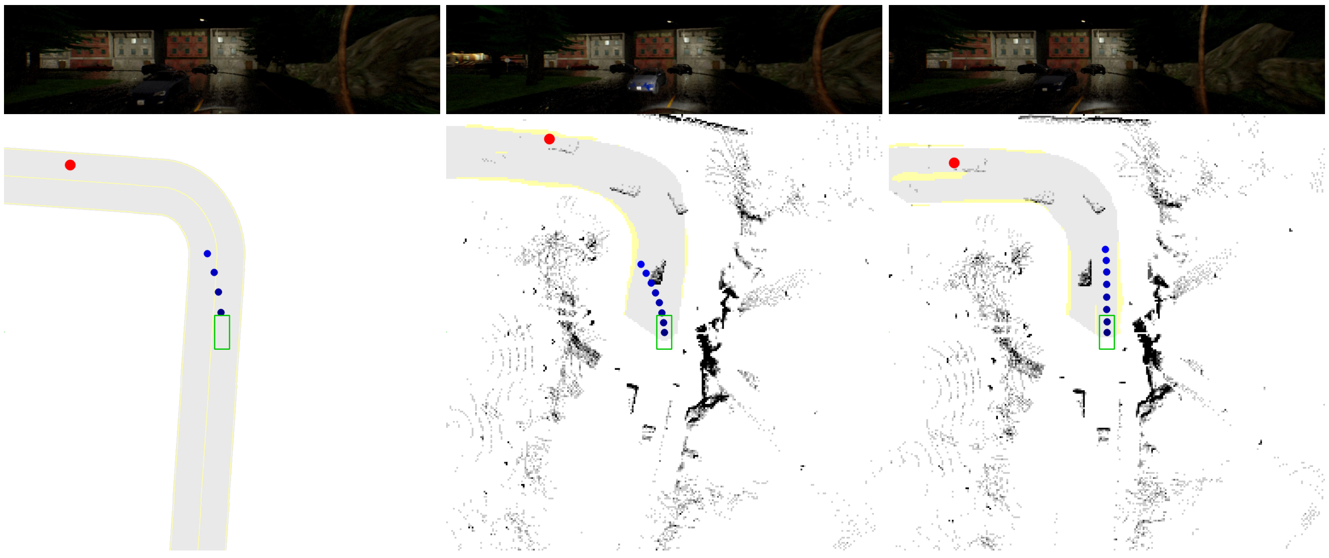

(a) TCP [31] follows a shortcut. (b) TransFuser follows a shortcut. (c) A transformer decoder fixes this.

Figure 3: **Target point shortcut.** When TP conditioned methods extrapolate to spatially distant waypoints, they incur large steering errors. Replacing global average pooling in TransFuser with a cross-attention mechanism mitigates the issue.

which indicates that it never takes a wrong turn or drives too far away (more than 30m) from the lane. However, this is not the case for NC conditioned models. While we do see a small improvement in DS with augmentation, its RC is still unsatisfactory. This indicates that the geometric information regarding the lane center available with the TP aids recovery in SotA IL methods, prompting further investigation. We inspect the TransFuser and LAV [8] models by constructing situations where the car is forcefully steered out of lane. Fig. 2 visualizes these scenarios with the camera, LiDAR and ground truth HD maps. For TP conditioned models, the predicted waypoints (shown in blue) extrapolate towards the nearest TP (red) even though the situation is out of distribution. This shows that one reason for their strong route following is that the bird's eye view (BEV) TP resets steering errors periodically: methods learn to steer towards nearby TPs because the expert trajectory in the dataset always goes through them. This has the effect that TP conditioned methods periodically drive back to the center of the lane, resetting any accumulated errors. Furthermore, it suggests that SotA IL approaches strongly rely on pre-specified geometric information regarding lane centers for recovery.

We identify this as a form of shortcut learning, which can be useful when the car is close to a target point, but can also lead to catastrophic steering errors when the target point is further away. An example would be directly extrapolating to a target point behind a turn, which leads to cutting the turn. In Fig. 3a and Fig. 3b, we show instances where this happens for TCP [31] and TransFuser [11] in a validation town at nighttime. TransFuser also predicts the BEV segmentation as an auxiliary task, which we overlay

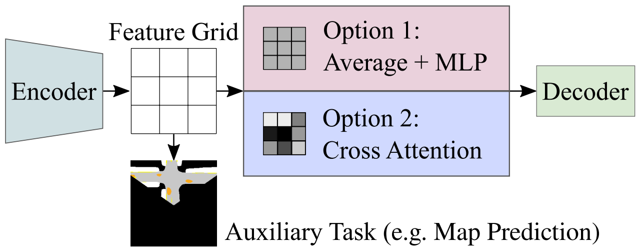


Figure 4: **Pooling.** Existing approaches vectorize feature grids either by global average pooling (top, e.g. [8, 11]) or with attention mechanisms (bottom, e.g. [26,31]). The latter retains spatial information gained via auxiliary tasks.

in the figure (gray: route, yellow: lane marking). For TCP, we render the ground truth map and omit the LiDAR, since it does not use LiDAR and does not predict BEV segmentation. Both methods predict waypoints that are tilted towards the TP instead of following the street. As a result, they drive into the opposing lane. We refer to this as the *TP shortcut*.

3.2. Improved pooling and data augmentation

One design choice which differs across SotA architectures is the pooling applied between the encoder and decoder. In Fig. 4, we summarize these. In particular, TransFuser [11] and LAV [8] employ global average pooling (GAP) followed by an MLP. InterFuser [26] pools features via the cross-attention mechanism of a transformer decoder. Finally, TCP [31], which has two decoders, uses GAP for one decoder and attention-based pooling for the other.

As shown in Fig. 4, the networks are encouraged to learn spatially meaningful feature grids through auxiliary convolutional decoders that predict outputs such as BEV semantic segmentation. However, GAP does not maintain the spatial information in the features. Fig. 3b shows the BEV road segmentation predicted by TransFuser overlaid by its LiDAR input. Unlike the waypoints, the BEV predictions are quite accurate. This indicates strong features coming from the encoder. Nevertheless, the final conditional waypoint predictions focus on the TP signal in this situation. We hypothesize that the GAP operation makes it difficult for the downstream decoder to utilize the strong BEV features. Note that attention-based pooling, such as the transformer decoder of Interfuser, preserves spatial information through the use of positional encodings.

We compare the original TransFuser GAP approach with the spatially preserving design of Interfuser in Table 2. For the latter variant, we remove the GAP operation and MLP in TransFuser. We then process the 8×8 BEV feature grid as tokens with a transformer decoder. Implementation details can be found in the supplementary material.

Pooling	**Aug.**	**DS** ↑	**RC** ↑	**Stat** ↓
GAP + MLP	-	39 ± 9	84 ± 7	1.04
Transformer Decoder	-	43 ± 6	**93** ± 3	0.55
Transformer Decoder	✓	**49** ± 8	90 ± 4	**0.10**

Table 2: **Pooling and augmentation.**

After replacing GAP with the transformer decoder, the RC increases by 9 points and collisions per km with static objects ("Stat") reduce by a factor of 2. Static objects (such as poles) only occur outside lanes in CARLA, implying that the network manages to stay in the lane far more consistently. Fig. 3c provides qualitative evidence that the transformer decoder can succeed in situations where GAP failed due to the TP shortcut (additional examples in supplementary). In addition, we investigate the inclusion of shift and rotation augmentations designed to aid lateral recovery (as described in Section 3.1). Collisions with the static environment are reduced further by a factor of 5 indicating that augmentation provides strong benefits.

3.3. The ambiguity of waypoints

Waypoints are used in many SotA systems as outputs [8, 11, 31]. They are obtained by recording an expert driver's GNSS locations at fixed time intervals (e.g. 0.5s) and transforming them into a local coordinate frame. The model is then trained to predict future waypoints, typically with an L_1 regression objective. Waypoints entangle both the path and the future velocities of the vehicle. The velocity after a specific time interval extracted from the waypoints is used by a downstream controller as a target speed.

In Fig. 5, we plot a histogram comparing the target speeds of the expert algorithm (i.e. training dataset) to those extracted from TransFuser (best model in Table 2). Interestingly, the distributions are quite different. By design, the expert chooses one of four target speeds: 29, 18, 7, or 0 km/h. These values cover behaviors needed for city driving, corresponding to four situations: regular driving, slowing down in intersections, slowing down near pedestrians, and stopping. In contrast, for TransFuser, the predicted target speeds cover the entire 0-29 km/h range.

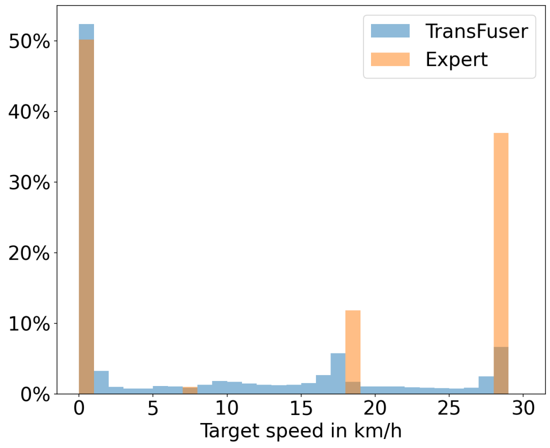
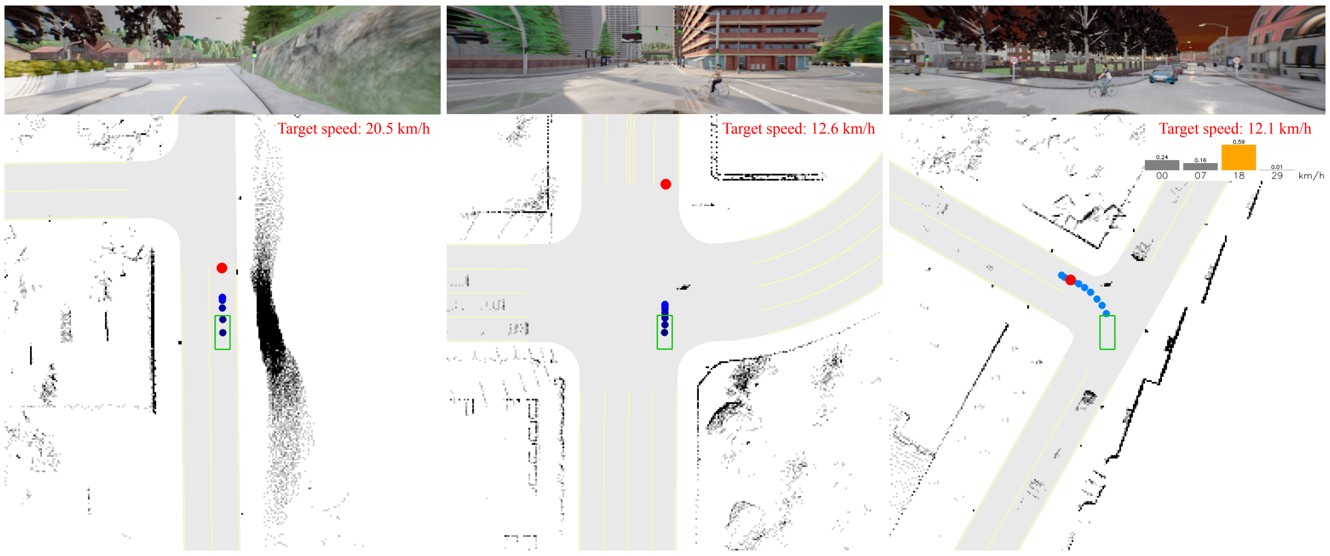


Figure 5: **Transfuser interpolates between modes.**

While the future path that our expert follows is deterministic and unambiguous (center of the lane, lane change at pre-defined locations), future velocities are multi-modal. As both are jointly represented by waypoints, this leads to an ambiguous entangled representation. However, existing methods which utilize this entangled waypoint representation predict only point estimates (i.e., a single set of waypoints) as output, hence only a single mode is modeled. From Fig. 5, we observe that the waypoint based TransFuser model indeed interpolates between modes, a behavior that is expected when modeling multi-modal outputs deterministically. We illustrate this behavior in detail using the examples of (1) approaching an intersection with a green light (Fig. 6a), and (2) a cyclist cutting into the vehicle's path (Fig. 6b). TransFuser slows down in these situations since it is uncertain, e.g., the light may turn red in Fig. 6a. The car stops in time in Fig. 6b because of the decreased speed.

While this averaging is indeed beneficial in some situations, an entangled representation is undesirable as it is less interpretable and does not explicitly expose uncertainty. Moreover, when stopped (all waypoints collapsed to one location), the steering signal is undefined which requires additional heuristics in the controller. To resolve this, we disentangle the future velocities from the path by sampling the expert's position at fixed distances instead of fixed time intervals for training a (deterministic) *path predictor*. However, when predicting the path instead of time-dependent trajectories, one requires an additional method to determine a target speed for the car. We propose to predict the tar-



(a) Slowing down at a green light. (b) Slowing down in an intersection. (c) Disentangling route and target speed.

Figure 6: **Waypoints are ambiguous.** The model's output representation forces it to predict a single mode for future velocities. There is a possibility that the traffic light might turn red in the future, or that the cyclist either cuts in or yields to the agent. The particular mode (or interpolation thereof) that the model converges to, depends on the training run and dataset.

get speed by simple classification using an additional MLP head. We incorporate the prediction uncertainties into the final output by using a confidence weighted average of the predicted target speed as input to the controller. Note that this is just one example of how the resulting confidence estimates may be leveraged by a vehicle controller.

Output	**DS** ↑	**RC** ↑	**Veh** ↓	**Stat** ↓
Waypoints	49 $\pm$ 8	**90** $\pm$ 4	**0.70**	0.10
Path + Argmax	40 $\pm$ 1	88 $\pm$ 2	1.25	0.03
Path + Weighted	**50** $\pm$ 3	88 $\pm$ 1	0.83	**0.02**

Table 3: **Output representation.**

Compared to the waypoint representation, naively converting the predicted classes to a target speed via the most likely class (Argmax) increases vehicle collisions ("Veh"), as seen in Table 3. Employing the proposed weighting (Weighted) achieves lower collisions. Moreover, the path-based model steers better as indicated by the lower static obstacle collisions ("Stat"). The disentangled representation achieves the same driving score as the entangled waypoint representation, providing an alternative with a simpler and more interpretable controller: (1) it eases design since it has identical parameters to the controller in the expert, (2) it has access to an unambiguous path representation even when driving slowly, and (3) it explicitly exposes and makes use of uncertainties with regard to target speeds.

Fig. 6c shows a qualitative example. The car is slowing down due to its uncertainty. The lower target speed allows it successfully merge behind the cyclist.

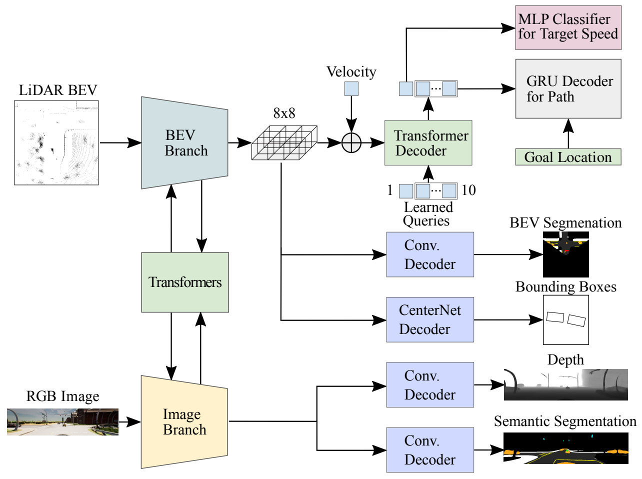


Figure 7: **TransFuser++ architecture.**

3.4. Scaling up to TransFuser++

By incorporating these insights, we obtain a significantly improved version of TransFuser [11] that we call TransFuser++ (Fig. 7). We now describe two important implementation changes compared to [11] that involve scaling.

Two stage	**Frozen**	**DS** ↑	**RC** ↑	**Veh** ↓
-	-	50 $\pm$ 3	88 $\pm$ 1	0.83
✓	-	**54** $\pm$ 1	92 $\pm$ 5	**0.79**
✓	✓	44 $\pm$ 6	**97** $\pm$ 1	1.21

Table 4: **Two stage training.**

Training schedule: We observe no benefits from simply doubling the number of training epochs for a single training stage. However, in Table 4, we double the training time with a two-stage approach. First, we pre-train the encoder with only the perception losses (2D depth, 2D and BEV semantic segmentation, and vehicle bounding boxes, as shown in Fig. 7) for the usual number of epochs. We then fine-tune the resulting checkpoint with all losses, after including the GRU decoder and MLP classifier. Initializing the backbone with features that are pre-trained on the auxiliary tasks leads to a 4 DS improvement, consistent with the claims of LAV [8]. We also experiment with freezing the pre-trained backbone and only training the transformer decoder and its heads in the second stage. This leads to a drop of 10 DS, indicating that end-to-end optimization is important.

Dataset scale: Recent work on CARLA [23] shows large improvements by scaling the dataset size. However, [23] considers planning models with privileged inputs on training towns. We investigate the impact of scaling when evaluating on held-out validation towns for end-to-end models. We start with 185k training samples (as in [11]) and scale it up by re-running the training routes 3 times with different traffic (555k frames). Table 5 shows a clear improvement of 6 DS via scaling. This shows that current IL approaches can still benefit from larger datasets. We train for the same amount of epochs, which implies that this improvement comes at the cost of 3× longer training.

Dataset size	DS ↑	RC ↑	Veh ↓
185k	54 ± 1	92 ± 5	0.79
555k	**60** ± 6	**98** ± 1	**0.73**

Table 5: **Effects of scale.**

4. Comparison to State of the Art

Methods: This section demonstrates the advantages of TF++ over several state-of-the-art models, which we list below, starting with the most recent. (1) **Interfuser** [26] processes multiple cameras and a BEV LiDAR by encoding each of them individually with a CNN. The resulting feature grids are fed into a transformer and decoded into perception outputs and the path to follow. The network does not predict longitudinal controls. Instead, it uses the BEV bounding boxes and a simple motion forecast (extrapolating historical dynamics) and linearly optimizes for the target speed using heuristically chosen objectives. (2) **Perception PlanT** [23] is a transformer based method trained in two independent stages that uses BEV bounding boxes as an intermediate representation. Its perception is based on TransFuser, and its planner outputs waypoints. (3) **TCP** [31] is a camera-only model with two output representations: waypoints and controls. During test time, it ensembles the two outputs together with a weighted average that changes based on whether the vehicle is turning. (4) **TransFuser** [11] fuses perspective cameras with a BEV LiDAR by processing them with individual CNNs and exchanging features using transformers. It uses global average pooling and waypoint outputs, as described in Section 3. (5) **LAV** [8] processes cameras and LiDAR point clouds into an intermediate BEV representation. It trains a planner on this representation that predicts waypoints for the ego vehicle with a NC conditioned GRU followed by a TP conditioned refinement GRU. During training, the planner is also tasked to predict the trajectory of other surrounding vehicles to increase the number of labels. The authors release two versions of this method, which we call LAV v1 and LAV v2. (6) **WOR** [7] is an IL method for which the labels are enriched using a reward function to provide dense supervision for all possible actions. Unlike the other baselines, it is conditioned with the NC and does not utilize the TP. WOR is the best NC conditioned baseline that is publicly available.

Benchmarks: We use two benchmarks to evaluate on seen and unseen towns, with scenarios taken from [11] (type 1,3,4,7,8,9 and 10). Table 6 compares the performance of systems on the Longest6 benchmark [11] which consists of 36 routes in training towns 01 to 06. On Longest6, models trained on any data can be evaluated, so we compare against the author-provided models or directly report numbers from the respective papers. The mean and std of three evaluations is reported. For TF++ we train 3 models and report the average result. Longest6 has the advantage that it evaluates in dense traffic and has diverse towns and routes. The drawbacks are that it does not penalize stop sign infractions or measure generalization to new towns. Therefore, we additionally compare performance on validation towns using the LAV [8] routes in Table 7. These are 4 routes with 4 weathers (16 combined) in Town 02 and 05, which are withheld during training. We re-train methods with the dataset from the corresponding paper 3 times and evaluate each seed 3 times. Reported results are the mean of all runs and the std between the training seeds. We compare against the reported SotA TCP and our reproduced TransFuser baseline on this benchmark and provide additional baselines in the supplementary.

Dataset: For TF++ we collect data on the same training routes as in [11] with an improved expert labeling algorithm (described in the supplementary material). We repeat this 3 times on the same routes with different traffic, as in [23]. For validation, we withhold Town 02 and 05 during training, else we train on all towns. In total, we train with 750k frames (550k when withholding Town 02 and 05).

Results: We start with the results in training towns (Table 6). The best NC conditioned method, WOR, has significantly lower RC than all TP conditioned systems (22 lower than LAV v1). In particular, the route deviations (Dev) are

Method	DS ↑	RC ↑	IS ↑	Ped ↓	Veh ↓	Stat ↓	Red ↓	Dev ↓	TO ↓	Block ↓
WOR [7]	21 ± 3	48 ± 4	0.56 ± 0.03	0.18	1.05	0.37	1.28	0.88	0.08	0.20
LAV v1 [8]	33 ± 1	70 ± 3	0.51 ± 0.02	0.16	0.83	0.15	0.96	0.06	0.12	0.45
Interfuser [26]	47 ± 6	74 ± 1	0.63 ± 0.07	0.06	1.14	0.11	0.24	**0.00**	0.52	**0.06**
TransFuser [11]	47 ± 6	93 ± 1	0.50 ± 0.06	0.03	2.45	0.07	0.16	**0.00**	**0.06**	0.10
TCP [31]	48 ± 3	72 ± 3	0.65 ± 0.04	0.04	1.08	0.23	0.14	0.02	0.18	0.35
LAV v2 [8]	58 ± 1	83 ± 1	0.68 ± 0.02	**0.00**	**0.69**	0.15	0.23	0.08	0.32	0.11
Perception PlanT [23]	58 ± 5	88 ± 1	0.65 ± 0.06	0.07	0.97	0.11	0.09	**0.00**	0.13	0.13
TF++ (ours)	**69** ± 0	**94** ± 2	**0.72** ± 0.01	**0.00**	0.83	**0.01**	**0.05**	**0.00**	0.07	**0.06**
Expert	*81* ± 3	*90* ± 1	*0.91* ± 0.04	*0.01*	*0.21*	*0.00*	*0.01*	*0.00*	*0.07*	*0.09*

Table 6: **Performance on training towns (Longest6).** Released models, std over 3 evaluations.

Method	DS ↑	RC ↑	IS ↑	Ped ↓	Veh ↓	Stat ↓	Red ↓	Stop ↓	Dev ↓	TO ↓	Block ↓
TransFuser (ours)	39 ± 9	84 ± 7	0.46 ± 0.06	**0.00**	0.74	1.04	0.20	1.07	0.00	0.23	0.21
TCP [31]	58 ± 5	85 ± 3	0.67 ± 0.06	**0.00**	**0.35**	0.16	**0.01**	1.05	0.00	0.19	0.19
TF++ (ours)	**70** ± 6	**99** ± 0	**0.70** ± 0.06	0.01	0.63	**0.01**	0.04	**0.26**	**0.00**	**0.05**	**0.00**
Expert	*94*	*95*	*0.99*	*0.00*	*0.02*	*0.00*	*0.02*	*0.00*	*0.00*	*0.00*	*0.08*

Table 7: **Performance on validation towns (LAV).** Reproduced models, std over 3 trainings (for 3 evaluations each).

10 times higher. Current TP conditioned SotA methods achieve (close to) 0 route deviations with driving scores in the range of 47 DS to 58 DS. TF++ outperforms all baselines on Longest6, showing a 19% relative improvement over the previous SotA Perception PlanT. TF++ achieves close to expert-level performance on all infractions except for vehicle collisions (Veh). On the validation towns (Table 7) TF++ outperforms our reproduced TransFuser by 31 DS improving all metrics, particularly environmental collisions (Stat). It improves the prior SotA result TCP by 21%.
Runtime: Table 8 compares the runtime of TF++ and TransFuser which have the same sensor setup. TF++ yields a large DS improvement while being only 14% slower.

Method ↑	DS ↑	Time (ms) ↓
TransFuser (ours)	39	44
TF++ (ours)	70	50

Table 8: **Runtime.** We show the runtime per frame in ms averaged over 300 time steps on a single route on a RTX 3090. TF++ outperforms TransFuser by a wide margin despite using a similar compute budget during inference.

5. Conclusion

In this work, we show that recent SotA driving models have exceptional route following abilities because they learn a strong bias towards following nearby TPs. Shortcut learning like this is a general phenomenon in deep neural networks [16] and has been observed in the context of autonomous driving for inputs such as velocity [14] or temporal frames [3, 30]. We add to this literature by observing shortcut learning with respect to the conditioning signal. While shortcut learning usually has a negative impact on performance, we observe positive (improved recovery) and negative (cutting turns) effects. We show that the negative effects can be mitigated by avoiding global pooling and incorporating data augmentation.

A second commonality in SotA approaches is the use of waypoints as an output representation. We observe that this representation is ambiguous because it predicts a point estimate for multi-modal future velocities. We disambiguate the representation by disentangling future velocities from the deterministic path predictions and classifying target speeds instead. We then weigh target speeds according to their predicted confidence in our controller. Surprisingly, we find that interpolation is helpful for reducing collisions.

We propose TransFuser++ by improving the popular baseline TransFuser with a series of controlled experiments. TF++ is a simple end-to-end method that sets a new state of the art on the LAV and Longest6 benchmarks.
Limitations: This study investigates urban driving in CARLA, where all investigated methods drive at relatively low speed ($<$ 35km/h). Therefore, problems specific to high-speed driving are not considered. In addition, lanes are free of static obstacles, hence scenarios requiring navigation around them are not included. We point out the strong reliance of current methods on TPs for recovery. This implies a reliance on accurate localization and mapping to obtain these TPs. They are accurately mapped in CARLA, however, this assumption might not hold in real environments.
Broader Impact: We aim to make progress towards autonomous driving. This technology, if realized, could have massive societal impact, reducing road accidents, trans-

portation costs and improving the mobility of elderly people. Potential negative implications include a reduction in jobs for human drivers and possible military applications.

Acknowledgements: Andreas Geiger and Bernhard Jaeger were supported by the ERC Starting Grant LEGO-3D (850533), the BMWi in the project KI Delta Learning (project number 19A19013O) 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 as well as Niklas Hanselmann and Katrin Renz for proofreading.

References

[1] Carla autonomous driving leaderboard. https://leaderboard.carla.org/, 2020. 1

[2] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022. 2

[3] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proc. Robotics: Science and Systems (RSS)*, 2019. 2, 8

[4] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2020. 3

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv.org*, 1604.07316, 2016. 2, 3

[6] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[7] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to drive from a world on rails. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 7, 8

[8] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3, 4, 5, 7, 8

[9] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Proc. Conf. on Robot Learning (CoRL)*, 2019. 1, 2

[10] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[11] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[12] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 2

[13] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2018. 1, 2, 3

[14] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2, 3, 8

[15] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017. 2

[16] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020. 1, 8

[17] Anthony Hu, Gianluca Corrado, Nicolas Griffiths, Zak Murez, Corina Gurau, Hudson Yeo, Alex Kendall, Roberto Cipolla, and Jamie Shotton. Model-based imitation learning for urban driving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[18] Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[19] Eshed Ohn-Bar, Aditya Prakash, Aseem Behl, Kashyap Chitta, and Andreas Geiger. Learning situational driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[20] Dean Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 1988. 2

[21] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[22] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2

[23] Katrin Renz, Kashyap Chitta, Otniel-Bogdan Mercea, Almut Sophia Koepke, Zeynep Akata, and Andreas Geiger. Plant: Explainable planning transformers via object-level

representations. In *Proc. Conf. on Robot Learning (CoRL)*, 2022. 2, 7, 8

[24] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 1, 3

[25] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Proc. Conf. on Robot Learning (CoRL)*, 2018. 2

[26] Hao Shao, Letian Wang, RuoBing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Proc. Conf. on Robot Learning (CoRL)*, 2022. 1, 2, 3, 4, 7, 8

[27] Hao Shao, Letian Wang, Ruobing Chen, Steven L. Waslander, Hongsheng Li, and Yu Liu. Reasonnet: End-to-end driving with temporal and global reasoning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[28] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017. 1

[30] Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 8

[31] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2, 3, 4, 5, 7, 8

[32] Jimuyang Zhang, Zanming Huang, and Eshed Ohn-Bar. Coaching a teachable student. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[33] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2