TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger

Abstract—How should we integrate representations from complementary sensors for autonomous driving? Geometry-based fusion has shown promise for perception (e.g. object detection, motion forecasting). However, in the context of end-to-end driving, we find that imitation learning based on existing sensor fusion methods underperforms in complex driving scenarios with a high density of dynamic agents. Therefore, we propose TransFuser, a mechanism to integrate image and LiDAR representations using self-attention. Our approach uses transformer modules at multiple resolutions to fuse perspective view and bird's eye view feature maps. We experimentally validate its efficacy on a challenging new benchmark with long routes and dense traffic, as well as the official leaderboard of the CARLA urban driving simulator. At the time of submission, TransFuser outperforms all prior work on the CARLA leaderboard in terms of driving score by a large margin. Compared to geometry-based fusion, TransFuser reduces the average collisions per kilometer by 48%.

Index Terms—Autonomous Driving, Imitation Learning, Sensor Fusion, Transformers, Attention.

1 Introduction

LiDAR sensors provide accurate 3D information for autonomous vehicles. While LiDAR-based methods have recently shown impressive results for end-to-end driving [1]–[4], they are evaluated in settings that assume access to privileged information not available through the LiDAR. This includes test-time access to HD maps and ground truth traffic light states. In practice, the information missing in the LiDAR must be recovered from other sensors on the vehicle, such as RGB cameras [5]–[13].

This raises important questions: *Can we integrate representations from these two modalities to exploit their complementary advantages for autonomous driving? To what extent should we process the different modalities independently, and what kind of fusion mechanism should we employ for maximum performance gain?* Prior works in the field of sensor fusion have mostly focused on the perception aspect of driving, e.g. 2D and 3D object detection [14]–[23], motion forecasting [14], [17], [24]–[31], and depth estimation [21], [22], [32], [33]. These methods focus on learning a state representation that captures the geometric and semantic information of the 3D scene. They operate primarily based on geometric feature projections between the image space and different LiDAR projection spaces, e.g. Bird's Eye View (BEV) [14]–[22] and Range View (RV) [14], [17], [23], [30], [34], [35]. Information is typically aggregated from a local neighborhood around each feature in the projected 2D or 3D space.

We observe that the locality assumption in these architecture designs hampers performance in complex urban scenarios (Table 1). For example, when handling traffic at intersections with multiple lanes, the ego-vehicle needs to account for interactions between nearby dynamic agents and traffic lights that are farther away. While deep convolutional networks can be used to capture global context within a single modality, it is non-trivial to extend them to multiple modalities or model interactions between pairs of features. To overcome these limitations, we use the attention mechanism of transformers [36], [37] to tightly integrate global contextual reasoning about the 3D scene directly into the feature extraction layers of different modalities. We con-

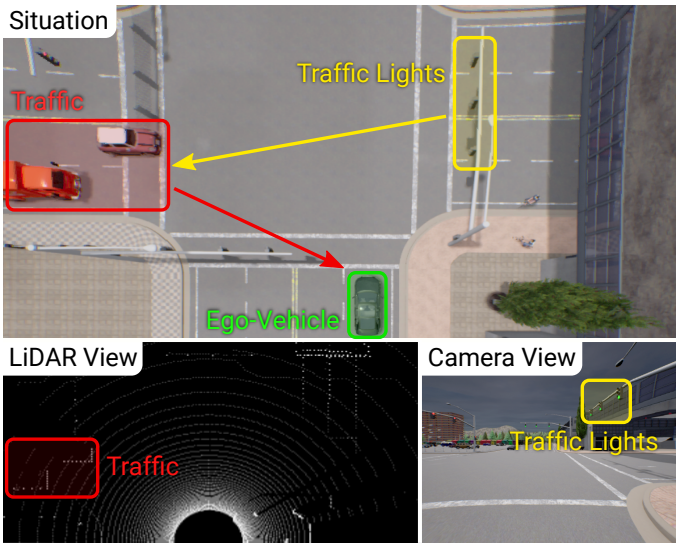


Fig. 1: **Illustration.** Consider an intersection with oncoming traffic from the left. To safely navigate the intersection, the agent (green) must capture the global context of the scene involving the interaction between the traffic light (yellow) and the crossing traffic (red). Our TransFuser model integrates geometric and semantic information across multiple modalities via attention mechanisms to capture global context, leading to safe driving behavior in CARLA.

- *K. Chitta, B. Jaeger, Z. Yu, K. Renz and A. Geiger are with the Autonomous Vision Group, University of Tübingen and Max Planck Institute for Intelligent Systems, Tübingen. E-mail: kashyap.chitta@tue.mpg.de, bernhard.jaeger@uni-tuebingen.de, zehao.yu@uni-tuebingen.de, katrin.renz@uni-tuebingen.de, a.geiger@uni-tuebingen.de.*
- *A. Prakash is with the University of Illinois Urbana-Champaign. Work done while at the Max Planck Institute for Intelligent Systems, Tübingen. E-mail: adityap9@illinois.edu*

sider image and LiDAR inputs since they are complementary to each other, and focus on integrating representations between these modalities (Fig. 1). The inputs are processed by two independent convolutional encoder branches, which are interconnected using transformers (Fig. 2). We call the resulting model *TransFuser* and integrate it into an auto-regressive waypoint prediction framework designed for end-to-end driving.

To show the advantages of our approach, we conduct a comprehensive study using the CARLA driving simulator [5]. We consider a more challenging evaluation setting than existing closed-loop driving benchmarks (e.g. NoCrash benchmark [38], NEAT routes [39]) based on the new CARLA version 0.9.10 leaderboard [40]. Our proposed *Longest6* benchmark involves ∼1.5km long routes, increased traffic density, and challenging pre-crash traffic scenarios. To tackle these challenges, we incorporate auxiliary supervision signals in a multi-task learning setup to train TransFuser and several strong baselines. On both the proposed benchmark and the secret routes of the official CARLA leaderboard, TransFuser achieves a significantly higher driving score than prior work.

Our contributions can be summarized as follows:

- We design a new evaluation setting in CARLA which demonstrates that imitation learning policies based on existing sensor fusion approaches are unable to handle challenging scenarios with dense traffic.
- We propose a novel multi-modal fusion transformer (TransFuser) to incorporate global context and pair-wise interactions into the feature extraction layers of different input modalities.
- We conduct a detailed empirical analysis demonstrating state-of-the-art driving performance with TransFuser on both the proposed evaluation setting and the official CARLA leaderboard. Our analysis provides insights and explores the current limitations of end-to-end driving models.

This journal paper is an extension of a conference paper published at CVPR 2021 [41]: we enhance the TransFuser model from [41] to obtain state-of-the-art performance by incorporating (1) an improved expert demonstrator for data collection, (2) a new sensor configuration with an increased field of view through multiple cameras, (3) an improved vision backbone architecture, and (4) an updated training procedure involving multi-task learning. We also provide a new image-only baseline, *Latent TransFuser*, which significantly outperforms prevalent baselines used for CARLA. Our updated code, dataset, and trained models are available at https://github.com/autonomousvision/transfuser.

2 Related Work

Multi-Modal Autonomous Driving: Recent multi-modal methods for end-to-end driving [35], [42]–[45] have shown that complementing RGB images with depth and semantics has the potential to improve driving performance. Xiao et al. [42] explore RGBD input from the perspective of early, mid and late fusion of camera and depth modalities and observe significant gains. Behl et al. [43] and Zhou et al. [44] demonstrate the effectiveness of semantics and depth as explicit intermediate representations for driving. Natan et al. [45] combine 2D semantics and depth into a semantic point cloud for end-to-end driving. In this work, we focus on image and LiDAR inputs since they are complementary to each other in terms of representing the scene and are readily available in autonomous driving systems. In this respect, Sobh et al. [35] exploit a late fusion architecture for LiDAR and image modalities where each input is encoded in a separate stream from which the final feature vectors are concatenated together. The concurrent work of Chen et al. [46] performs sensor fusion via PointPainting, which concatenates semantic class information extracted from the RGB image to the LiDAR point cloud [47]. We observe that the late fusion mechanism of Sobh et al. [35] suffers from high infraction rates and the PointPainting-based system has a reduced route completion percentage in comparison to several baselines (Table 1). To mitigate these limitations, we propose a multi-modal fusion transformer (TransFuser) that is effective in integrating information from different modalities at multiple stages during feature encoding using attention. TransFuser helps to capture the global context of the 3D scene which requires fusing information from distinct spatial locations across sensors.

Sensor Fusion Methods for Object Detection and Motion Forecasting: Most sensor fusion works consider perception tasks, e.g. object detection [14]–[16], [18]–[23], [47]–[60] and motion forecasting [24]–[30], [49], [61], [62]. They operate on multi-view LiDAR, e.g. Bird's Eye View (BEV) and Range View (RV), or complement the camera input with depth information from LiDAR. This is typically achieved by projecting LiDAR features into the image space or projecting image features into the BEV or RV space. The closest approach to ours is ContFuse [20] which performs multi-scale dense feature fusion between image and LiDAR BEV features. For each pixel in the LiDAR BEV representation, it computes the nearest neighbors in a local neighborhood in 3D space, projects these neighboring points into the image space to obtain the corresponding image features, aggregates these features using continuous convolutions, and combines them with the LiDAR BEV features. Concurrent to our work, EPNET++ [63] and CAT-Det [64] also employ multi-scale bidirectional fusion between image and LiDAR point clouds using attention to learn enhanced feature representations for 3D object detection. Other projection-based fusion methods follow a similar trend and aggregate information from a local neighborhood in 2D or 3D space. However, the state representation learned by these methods is insufficient since they do not capture the global context of the 3D scene, which is important for safe maneuvers in dense traffic. To demonstrate this, we implement a multi-scale geometry-based fusion mechanism, inspired by [20], [22], involving both image-to-LiDAR and LiDAR-to-image feature fusion for end-to-end driving and observe high infraction rates in the dense urban setting (Table 1).

Attention for Autonomous Driving: Attention has been explored in the context of driving for lane changing [65], object detection [31], [66]–[68], motion forecasting [31], [69]–[77], driver attention prediction [78], [79] and recently also for end-to-end driving [39], [41]. Chen et al. [66] employ a recurrent attention mechanism over a learned semantic map

for predicting vehicle controls. Li et al. [31] utilize attention to capture temporal and spatial dependencies between actors by incorporating a transformer module into a recurrent neural network. SA-NMP [75] learns an attention mask over features extracted from a 2D CNN, operating on LiDAR BEV projections and HD maps, to focus on dynamic agents for safe motion planning. Chen et al. [65] utilize spatial and temporal attention in a hierarchical deep reinforcement learning framework to focus on the surrounding vehicles for lane changing in the TORCS simulator. PYVA [80] uses attention to perform image translation from the perspective view to the BEV. This idea has also been adopted by several concurrent papers on BEV semantic segmentation from image inputs [81]–[85]. NEAT [39] uses intermediate attention maps to iteratively compress high dimensional 2D image features into a compact BEV representation for driving. Compared to NEAT, our attention mechanism is simpler since it does not require iterative refinement of the attention at test-time. Unlike NEAT, we also apply our attention mechanism at multiple feature resolutions, enabling sensor fusion for both shallow and deep features in the network. Furthermore, none of the existing attention-based driving approaches consider multiple sensor modalities. Our work uses the self-attention mechanism of transformers for dense fusion of image and LiDAR features.

3 TransFuser

In this work, we propose a novel architecture for end-to-end driving (Fig. 2). It has two main components: (1) a multi-modal fusion transformer for integrating information from multiple sensor modalities (image and LiDAR), and (2) an auto-regressive waypoint prediction network. The following sections detail our problem setting, input and output parameterization, and each component of the model.

3.1 Problem Setting

We consider the task of point-to-point navigation in an urban setting [1], [2], [38], [86], [87] where the goal is to complete a given route while safely reacting to other dynamic agents and following traffic rules.

Imitation Learning (IL): The goal of IL is to learn a policy π that imitates the behavior of an expert π^*. In our setup, a policy is a mapping from inputs to waypoints that are provided to a separate low-level controller to output actions. We consider the Behavior Cloning (BC) approach of IL which is a supervised learning method. An expert policy is first rolled out in the environment to collect a dataset, $\mathcal{D} = \{(\mathcal{X}^i, \mathcal{W}^i)\}_{i=1}^{Z}$ of size Z, which consists of high-dimensional observations of the environment, $\mathcal{X}$, and the corresponding expert trajectory, defined by a set of 2D waypoints in BEV space, i.e., $\mathcal{W} = \{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^{T}$. This BEV space uses the coordinate frame of the ego-vehicle. The policy, π, is trained in a supervised manner using the collected data, $\mathcal{D}$, with the loss function, $\mathcal{L}$.

$$\underset{\pi}{\operatorname{argmin}} \; \mathbb{E}_{(\mathcal{X},\mathcal{W})\sim\mathcal{D}} \left[\mathcal{L}(\mathcal{W}, \pi(\mathcal{X}))\right] \tag{1}$$

The high-dimensional observation, $\mathcal{X}$, includes a front camera image input and a LiDAR point cloud from a single time-step. We use a single time-step input since prior works on IL for autonomous driving have shown that using observation histories may not lead to performance gain [88]–[92]. For $\mathcal{L}$, we use the L_1 distance between the predicted trajectory, $\pi(\mathcal{X})$, and the expert trajectory, $\mathcal{W}$, as the primary loss function. Furthermore, we use several auxiliary losses used to boost performance, which are detailed in Section 3.6. We assume access to an inverse dynamics model [93], implemented as a PID Controller $\mathbb{I}$, which performs the low-level control, i.e., steer, throttle, and brake, provided the future trajectory $\mathcal{W}$. The actions are determined as $\mathbf{a} = \mathbb{I}(\mathcal{W})$.

Global Planner: We follow the standard protocol of CARLA 0.9.10 and assume that high-level goal locations $\mathcal{G}$ are provided as GPS coordinates. Note that these goal locations are sparse and can be hundreds of meters apart, as opposed to the local waypoints predicted by the policy π.

3.2 Input and Output Parameterization

Input Representation: Following previous LiDAR-based driving approaches [2], [86], we convert the LiDAR point cloud into a 2-bin histogram over a 2D BEV grid with a fixed resolution. We consider the points within 32m in front of the ego-vehicle and 16m to each of the sides, thereby encompassing a BEV grid of 32m $\times$ 32m. We divide the grid into blocks of 0.125m $\times$ 0.125m which results in a resolution of 256 $\times$ 256 pixels. For the histogram, we discretize the height dimension into 2 bins representing the points on/below and above the ground plane. We also rasterize the 2D goal location in the same 256 $\times$ 256 BEV space as the LiDAR point cloud and concatenate this channel to the 2 histogram bins. This results in a three-channel pseudo-image of size 256×256 pixels. We represent the goal location in the BEV as this correlates better with the waypoint predictions compared to the perspective image domain [87].

For the RGB input, we use three cameras (facing forward, 60° left and 60° right). Each camera has a horizontal FOV of 120°. We extract the images at a resolution of 960 $\times$ 480 pixels, which we crop to 320 $\times$ 160 to remove radial distortion at the edges. These three undistorted images are composed into a single image input to the encoder, which has a resolution of 704 $\times$ 160 pixels and 132° FOV. We find that this FOV is sufficient to observe both near and far traffic lights in all public towns of CARLA.

Output Representation: We predict the future trajectory $\mathcal{W}$ of the ego-vehicle in BEV space, centered at the current coordinate frame of the ego-vehicle. The trajectory is represented by a sequence of 2D waypoints, $\{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^{T}$. We use $T = 4$, which is the default number of waypoints required by our inverse dynamics model.

3.3 Multi-Modal Fusion Transformer

Our key idea is to exploit the self-attention mechanism of transformers [36] to incorporate the global context for image and LiDAR modalities, given their complementary nature. The transformer architecture takes as input a sequence consisting of discrete tokens, each represented by a feature vector. The feature vector is supplemented by a positional encoding to incorporate spatial inductive biases.

Formally, we denote the input sequence as $\mathbf{F}^{in} \in \mathbb{R}^{N \times D_f}$, where N is the number of tokens in the sequence,

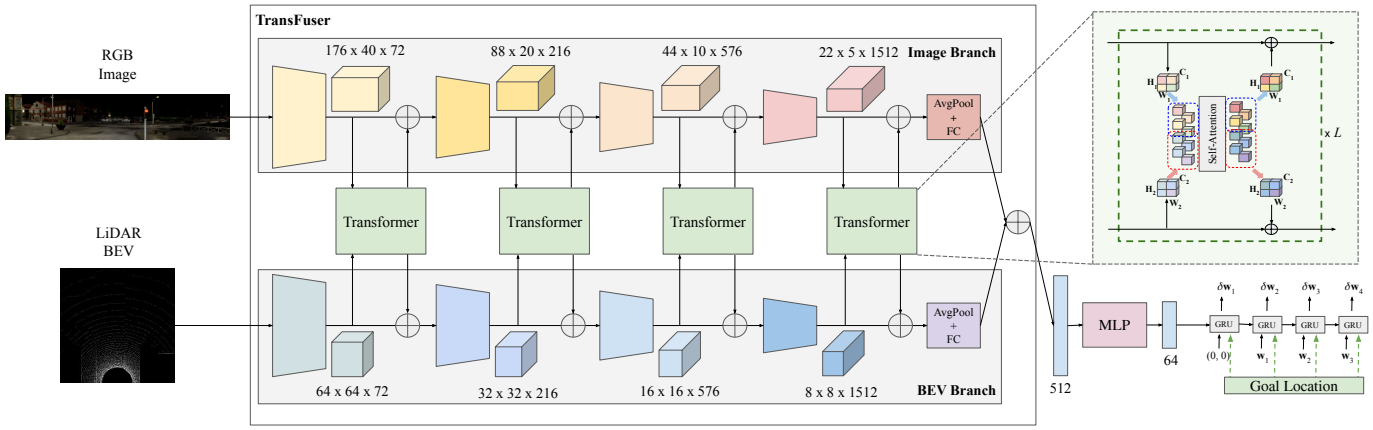


Fig. 2: **Architecture.** We consider RGB image and LiDAR BEV representations (Section 3.2) as inputs to our multi-modal fusion transformer (TransFuser) which uses several transformer modules for the fusion of intermediate feature maps between both modalities. This fusion is applied at multiple resolutions throughout the feature extractor, resulting in a 512-dimensional feature vector output from both the image and LiDAR BEV stream, which are combined via element-wise summation. This 512-dimensional feature vector constitutes a compact representation of the environment that encodes the global context of the 3D scene. It is then processed with an MLP before passing it to an auto-regressive waypoint prediction network. We use a single layer GRU followed by a linear layer that takes in the hidden state and predicts the differential ego-vehicle waypoints $\{\delta \mathbf{w}_t\}_{t=1}^{T}$, represented in the ego-vehicle's current coordinate frame.

and each token is represented by a feature vector of dimensionality D_f. The transformer uses linear projections for computing a set of queries, keys, and values ($\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$),

$$\mathbf{Q} = \mathbf{F}^{in}\mathbf{M}^{q}, \;\; \mathbf{K} = \mathbf{F}^{in}\mathbf{M}^{k}, \;\; \mathbf{V} = \mathbf{F}^{in}\mathbf{M}^{v} \tag{2}$$

where $\mathbf{M}^{q} \in \mathbb{R}^{D_f \times D_q}$, $\mathbf{M}^{k} \in \mathbb{R}^{D_f \times D_k}$ and $\mathbf{M}^{v} \in \mathbb{R}^{D_f \times D_v}$ are weight matrices. It uses the scaled dot products between $\mathbf{Q}$ and $\mathbf{K}$ to compute the attention weights and then aggregates the values for each query,

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{D_k}}\right)\mathbf{V} \tag{3}$$

Finally, the transformer uses a non-linear transformation to calculate the output features, $\mathbf{F}^{out}$ which are of the same shape as the input features, $\mathbf{F}^{in}$.

$$\mathbf{F}^{out} = \text{MLP}(\mathbf{A}) + \mathbf{F}^{in} \tag{4}$$

The transformer applies the attention mechanism multiple times throughout the architecture, resulting in L attention layers. Each layer in a standard transformer has multiple parallel attention 'heads', which involve generating several $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ values per $\mathbf{F}^{in}$ for Eq. (2) and concatenating the resulting values of $\mathbf{A}$ from Eq. (3).

Unlike the token input structures in NLP, we operate on grid structured feature maps. Similar to prior works on the application of transformers to images [94]–[97], we consider the intermediate feature maps of each modality to be a set rather than a spatial grid and treat each element of the set as a token. The convolutional feature extractors for the image and LiDAR BEV inputs encode different aspects of the scene at different layers. Therefore, we fuse these features at multiple scales (Fig. 2) throughout the encoder.

Let the intermediate grid structured feature map of a single modality indexed by s be a 3D tensor of dimension $H_s \times W_s \times C$. For S different modalities, these features are stacked together to form a sequence of dimension $\sum_{s=1}^{S}(H_s * W_s) \times C$. We add a learnable positional embedding, which is a trainable parameter of the same dimension as the stacked sequence, so that the network can infer spatial dependencies between different tokens at train time. The input sequence and positional embedding are combined using element-wise summation to form a tensor of dimension $\sum_{s=1}^{S}(H_s * W_s) \times C$. As shown in Fig. 2, this tensor is fed as input to the transformer, which produces an output of the same dimension. We have omitted the positional embedding and velocity embedding inputs in Fig. 2 for clarity. The output is then reshaped into S feature maps of dimension $H_s \times W_s \times C$ each and fed back into each of the individual modality branches using an element-wise summation with the existing feature maps. The mechanism described above constitutes feature fusion at a single scale. This fusion is applied *multiple times* throughout the feature extractors of the image and BEV branches at different resolutions (Fig. 2). However, processing feature maps at high spatial resolutions is computationally expensive. Therefore, we downsample higher resolution feature maps from the early encoder blocks using average pooling to the same resolution as the final feature map before passing them as inputs to the transformer. We upsample the output to the original resolution using bilinear interpolation before element-wise summation with the existing feature maps.

After carrying out dense feature fusion at multiple resolutions (Fig. 2), we obtain a feature map of dimensions $22 \times 5 \times C$ from the feature extractors of the image branch, and $8 \times 8 \times C$ from the BEV branch. Where C is the number of channels at the current resolution in the feature extractor and lies in $\{72, 216, 576, 1512\}$. These feature maps are reduced to a dimension of 512 by average pooling, followed by a fully-connected layer of 512 units. The feature vector of dimension 512 from both the image and the LiDAR BEV streams are then combined via element-wise summation.

This 512-dimensional feature vector constitutes a compact representation of the environment that encodes the global context of the 3D scene. This is then fed to the waypoint prediction network, which we describe next.

3.4 Waypoint Prediction Network

As shown in Fig. 2, we pass the 512-dimensional feature vector through an MLP (comprising 2 hidden layers with 256 and 128 units) to reduce its dimensionality to 64 for computational efficiency before passing it to the auto-regressive waypoint network implemented using GRUs [98]. We initialize the hidden state of the GRU with the 64-dimensional feature vector. The update gate of the GRU controls the flow of information encoded in the hidden state to the output and the next time-step. It also takes in the current position and the goal location (Section 3.1) as input, which allows the network to focus on the relevant context in the hidden state for predicting the next waypoint. We provide the GPS coordinates of the goal location (registered to the ego-vehicle coordinate frame) as input to the GRU in addition to the encoder since it can more directly influence the waypoint predictions. Following [2], we use a single layer GRU followed by a linear layer which takes in the hidden state and predicts the differential ego-vehicle waypoints $\{\delta\mathbf{w}_t\}_{t=1}^{T}$ for $T = 4$ future time-steps in the ego-vehicle current coordinate frame. Therefore, the predicted future waypoints are given by $\{\mathbf{w}_t = \mathbf{w}_{t-1} + \delta\mathbf{w}_t\}_{t=1}^{T}$. The input to the first GRU unit is given as (0,0) since the BEV space is centered at the ego-vehicle's position.

3.5 Controller

We use two PID controllers for lateral and longitudinal control to obtain steer, throttle, and brake values from the predicted waypoints, $\{\mathbf{w}_t\}_{t=1}^{T}$. The longitudinal controller takes in the magnitude of a weighted average of the vectors between waypoints of consecutive time steps, whereas the lateral controller takes in their orientation. For the PID controllers, we use the same configuration as in the author-provided codebase of [87]. Additional details regarding the controllers can be found in the supplementary material.

Creeping: If the car has not moved for a long duration (55 seconds, chosen to be higher than the expected wait time at a red light), we make it creep forward by setting the target speed of the PID controller to 4 m/s for a short time (1.5 seconds). This creeping behavior is used to recover from the *inertia problem* observed in IL for autonomous driving [38]. When a vehicle is still, the probability that it continues to stay in place (e.g. in dense traffic) is very high in the training data. This can lead to the trained agent never starting to drive again after having stopped.

Safety Heuristic: The creeping behavior alone would be unsafe, e.g. in situations where the agent is stuck in traffic where creeping forward could lead to a collision. To prevent this, we implement a safety check that overwrites the creeping behavior if there are any LiDAR hits in a small rectangular area in front of the car. While this heuristic is essential during creeping, it can also be applied during regular driving to enhance the safety [99]. We study the impact of applying a global safety heuristic during both creeping and regular driving in Section 4.11.

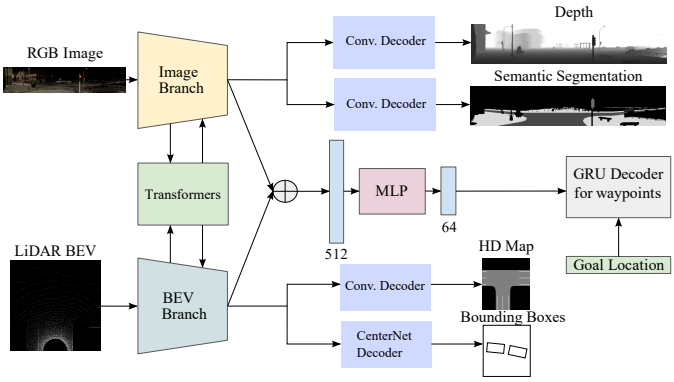


Fig. 3: **Auxiliary Loss Functions.** Besides the waypoint loss (Eq. (5)), we incorporate four auxiliary tasks: depth prediction and semantic segmentation from the image branch; HD map prediction and vehicle detection from the BEV branch.

3.6 Loss Functions

Following [87], we train the network using an L_1 loss between the predicted waypoints and the ground truth waypoints (from the expert), registered to the current coordinate frame. Let $\mathbf{w}_t^{gt}$ represent the ground truth waypoint for time-step t, then the loss function is given by:

$$\mathcal{L} = \sum_{t=1}^{T} ||\mathbf{w}_t - \mathbf{w}_t^{gt}||_1 \tag{5}$$

Note that the ground truth waypoints $\{\mathbf{w}_t^{gt}\}$ which are available only at training time are different from the sparse goal locations $\mathcal{G}$ provided at both training and test time.

Auxiliary Tasks: To account for the complex spatial and temporal scene structure encountered in autonomous driving, the training objectives used in IL-based driving agents have evolved by incorporating auxiliary tasks. Training signals aiming to reconstruct the scene have become common, such as image auto-encoding [100], 2D semantic segmentation [101], Bird's Eye View (BEV) semantic segmentation [102], 2D semantic prediction [103], and BEV semantic prediction [3], [39]. Performing auxiliary tasks has been shown to lead to more interpretable and robust models [3], [39]. In this work, we consider four auxiliary tasks: depth prediction, semantic segmentation, HD map prediction, and vehicle detection (Fig. 3).

2D Depth and Semantics: Combining 2D depth estimation and 2D semantic segmentation as auxiliary tasks has been an effective approach for image-based end-to-end driving [39], [104]. We use the same decoder architecture as the AIM-MT baseline of [39] to decode depth and semantics from the image branch features. The depth output is supervised with an L_1 loss, and the semantics with a cross-entropy loss. Following [39], we consider 7 semantic classes: (1) unlabeled, (2) vehicle, (3) road, (4) red light, (5) pedestrian, (6) lane marking, and (7) sidewalk.

HD Map: We predict a three-channel BEV segmentation mask containing the classes road, lane marking and other. This encourages the intermediate features to encode information regarding drivable and non-drivable areas. The map

uses the same coordinate frame as the LiDAR input, and is therefore obtained from the feature map of the LiDAR branch with a convolutional decoder. However, we predict a downsampled version of the HD map (64×64 instead of 256×256) for computational efficiency. The HD map prediction task uses a cross-entropy loss.

Bounding Boxes: We locate other vehicles in the scene via keypoint estimation with a CenterNet decoder [105]. Specifically, we predict a position map $\hat{\mathbf{P}} \in [0,1]^{64\times64}$ from the BEV features using a convolutional decoder. Similar to the HD map prediction task, the 256×256 input is mapped to smaller 64×64 predictions for vehicle detection. The 2D target label for this task is rendered with a Gaussian kernel at each object center of our training dataset. While the orientation is a single scalar value, directly regressing this value is challenging, as observed in existing 3D detectors [105]–[107]. Therefore, our CenterNet implementation takes a two-stage approach of predicting an initial coarse orientation followed by a fine offset angle. To predict the coarse orientation, we discretize the relative yaw of each ground truth vehicle into 12 bins of size 30°, and predict this class via a 12-channel classification label at each pixel, $\hat{\mathbf{O}} \in [0,1]^{64\times64\times12}$, as in [107]. Finally, we predict a regression map $\hat{\mathbf{R}} \in \mathbb{R}^{64\times64\times5}$. This regression map holds three regression targets: vehicle size ($\in \mathbb{R}^2$), position offset ($\in \mathbb{R}^2$) and orientation offset ($\in \mathbb{R}$). The position offset is used to make up for quantization errors introduced by predicting position maps at a lower resolution than the inputs. The orientation offset corrects the orientation discretization error [107]. Note that only locations with vehicle centers are supervised for predicting $\hat{\mathbf{O}}$ and $\hat{\mathbf{R}}$. The position map, orientation map and regression map use a focal loss [108], cross-entropy loss, and L_1 loss respectively.

3.7 Latent TransFuser

CILRS [38] is a widely used image-only baseline for the old CARLA version 0.8.4. It learns to directly predict vehicle controls (as opposed to waypoints) from visual features while being conditioned on a discrete navigational command (follow lane, change lane left/right, turn left/right). However, as shown in [39] this approach obtains poor results on the challenging CARLA leaderboard. Despite this, recent studies involving image-based driving on CARLA have shifted from IL towards more complex Reinforcement Learning (RL) based training, while using CILRS as the primary IL baseline [109], [110]. To provide a more meaningful IL baseline for future studies, we now introduce an image-only version of our approach, called Latent TransFuser.

Latent TransFuser replaces the 2-channel LiDAR BEV histogram input to our architecture with a 2-channel positional encoding of identical dimensions ($256 \times 256 \times 2$). The 2D positional encoding is a grid with equally-spaced values from -1 to 1, with one channel for the left-right axis, and one for the top-down axis. Other than this change, the architecture, training procedure, and auxiliary losses remain identical to the LiDAR-based TransFuser. Additionally, our controller uses the LiDAR input for its safety heuristic while creeping (Section 3.5). For Latent TransFuser, we check for an intersection between the small rectangular safety area in front of the car and any detected object from the auxiliary CenterNet detection head instead. The creeping is disabled whenever such an intersection occurs.

The positional encoding input of Latent TransFuser acts as a proxy for the BEV LiDAR. Since the LiDAR branch is supervised to predict the HD map and bounding boxes, which are in the BEV coordinate frame, fusing image features with the latent features in this branch acts as an attention-based projection from the perspective view to the BEV. The architecture shares similarities with existing attention-based camera to BEV projection techniques such as NEAT [39] and PYVA [80]. However, unlike these methods, Latent TransFuser projects features at multiple feature resolutions. We show in our experiments that Latent TransFuser is a powerful baseline, outperforming far more complex RL-based methods on the CARLA leaderboard in the image-only input setting (Table 3).

4 Experiments

In this section, we describe our experimental setup, compare the **driving performance** of TransFuser against several baselines, visualize the **attention maps** of TransFuser and present **ablation studies** to highlight the importance of different components of our approach.

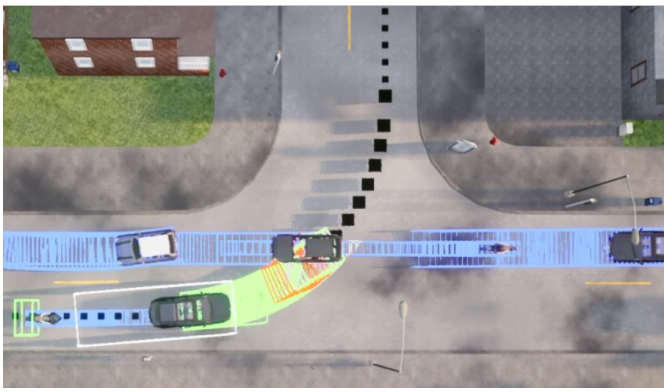
4.1 Task

We consider the task of navigation along a set of predefined routes in a variety of areas, e.g. freeways, urban areas, and residential districts. The routes are defined by a sequence of sparse goal locations in GPS coordinates provided by a global planner. Each route consists of several scenarios, initialized at predefined positions, which test the ability of the agent to handle different kinds of adversarial situations, e.g. obstacle avoidance, unprotected turns at intersections, vehicles running red lights, and pedestrians emerging from occluded regions to cross the road at random locations. The agent needs to complete the route within a specified time limit while following traffic regulations and coping with high densities of dynamic agents.

4.2 Training Dataset

We use the CARLA [5] simulator for training and testing, specifically CARLA 0.9.10 which consists of 8 publicly available towns. We use all 8 towns for training. Our dataset is collected along a set of training routes: around 2500 routes through junctions with an average length of 100m, and around 1000 routes along curved highways with an average length of 400m. For generating training data, we roll out an expert policy designed to drive using privileged information from the simulation and store data at 2FPS. The expert is a rule-based algorithm similar to the CARLA traffic manager autopilot[1]. Our training dataset contains 228k frames in total. In the following, we provide more details regarding the expert algorithm.

1. https://carla.readthedocs.io/en/latest/adv_traffic_manager/



(a) The expert waits before taking the turn because the trajectory forecasting predicts a collision if the expert would drive.

(b) After the oncoming cars have passed, the expert crosses the intersection.

Fig. 4: **Expert performing an unprotected left turn.** The black boxes on the street mark the path that the expert has to follow. The predictions of the bicycle model are colored green for the expert and blue for all other vehicles. Red bounding boxes mark predicted collisions. The white box around the car is used to detect the traffic light trigger boxes that are placed on the street (e.g. bottom left Fig. 4a).

4.3 Expert

For generating training data, we roll out an expert policy designed to drive using privileged information from the simulator. The waypoints of the expert are the ground truth labels for the imitation loss, so it can be viewed as an automatic labeling algorithm. The ground truth labels for the auxiliary tasks are provided by the simulator. We build upon the code provided by the authors of [87]. This approach is based on simple handcrafted rules. Building the expert with RL is also possible [111], [112] but it is more computationally demanding and less interpretable. Our expert policy consists of an A* planner followed by 2 PID controllers (for lateral and longitudinal control). The lateral and longitudinal control tasks are treated independently.

Lateral control is done by following the path generated by the A* planner. Specifically, we minimize the angle of the car towards the next waypoint in the route, which is at least 3.5 meters away, using a PID controller. Longitudinal control is done using a version of model predictive control and differentiates between 3 target speeds. The standard target speed is 4.0 m/s. When the expert is inside an intersection, the target speed is reduced to 3.0 m/s. Lastly, in case an infraction is predicted the target speed is set to 0.0 m/s bringing the vehicle to a halt. We predict red light infractions by performing intersection tests with trigger boxes that CARLA provides. Collision infractions are predicted by forecasting the oriented bounding boxes of all traffic participants. We forecast in 50 ms discrete steps, for 4 seconds in intersections and 1 second in other areas. The forecasting is done using the pretrained kinematic bicycle model from [110]. This bicycle model is a simple mathematical model that can predict the position, orientation, and speed of a car after a discrete time step, given its current position orientation speed and the applied control. We forecast all vehicles by iteratively rolling out the bicycle model using its output at time step t as the input for time step $t+1$. Since we only know the control input of other traffic participants at the current time step (provided by the simulator), we assume that they will continue to apply the same control at future time steps. For the ego vehicle, we calculate its future steering and throttle by using PID controllers that try to follow the route. The ego brake control is always set to 0 because we want to answer the counterfactual question of whether there will be a collision if we do not brake. We forecast pedestrians analogously but model them as a point with velocity and acceleration. This works well because the movement patterns of pedestrians in CARLA are simple. A collision infraction is detected if there is an intersection of the ego vehicle bounding box at future time step t with the bounding box of another traffic participant at future time step t. A common failure of the action repeat forecast mechanism described above is that it does not anticipate that fast cars approaching the expert from behind will eventually slow down before colliding. To avoid false positives, we do not consider rear-end collisions by only using the front half of the vehicle as its bounding box. For the longitudinal controller, we set $K_p = 5.0, K_i = 0.5, K_d = 1.0$ and for the lateral controller, we set $K_p = 1.25, K_i = 0.75, K_d = 0.3$. Both controllers use a buffer of size 40 to approximate the integral term as a running average. An example of the expert performing an unprotected left turn can be seen in Fig. 4.

4.4 Longest6 Benchmark

The CARLA simulator provides an official evaluation leaderboard consisting of 100 secret routes. However, teams using this platform are restricted to only 200 hours of evaluation time per month. A single evaluation takes over 100 hours, making the official leaderboard unsuitable for ablation studies or obtaining detailed statistics involving multiple evaluations of each model. Therefore, we propose the Longest6 Benchmark, which shares several similarities to the official leaderboard, but can be used for evaluation on local resources without computational budget restrictions.

The CARLA leaderboard repository provides a set of 76 routes as a starting point for training and evaluating agents. These routes were originally released along with the 2019 CARLA challenge. They span 6 towns and each of them is defined by a sequence of waypoints. However, there is a large imbalance in the number of routes per town, e.g. Town03 and Town04 have 20 routes each, but Town02 has only 6 routes. To balance the Longest6 driving benchmark

across all available towns, we choose the 6 longest routes per town from the set of 76 routes. This results in 36 routes with an average route length of 1.5km, which is similar to the average route length of the official leaderboard (1.7km). We make three other design choices for the Longest6 benchmark, motivated by the official leaderboard. (1) During evaluation, we ensure a high density of dynamic agents by spawning vehicles at every possible spawn point permitted by the CARLA simulator. (2) Following [39], each route has a unique environmental condition obtained by combining one of 6 weather conditions (Cloudy, Wet, MidRain, WetCloudy, HardRain, SoftRain) with one of 6 daylight conditions (Night, Twilight, Dawn, Morning, Noon, Sunset). (3) We include CARLA's adversarial scenarios in the evaluation, which are spawned at predefined positions along the route. Specifically, we include CARLA's scenarios 1, 3, 4, 7, 8, 9, 10 which are generated based on the NHTSA pre-crash typology[2]. Visualizations of the routes in the Longest6 benchmark are provided in the supplementary material.

4.5 Metrics

We report several metrics to provide a comprehensive understanding of the driving behavior of each agent.

(1) **Route Completion (RC)**: percentage of route distance completed, R_i by the agent in route i, averaged across N routes. However, if an agent drives outside the route lanes for a percentage of the route, then the RC is reduced by a multiplier (1- % off route distance).

$$\text{RC} = \frac{1}{N}\sum_{i}^{N} R_i \tag{6}$$

(2) **Infraction Score (IS)**: geometric series of infraction penalty coefficients, p^j for every instance of infraction j incurred by the agent during the route. Agents start with an ideal 1.0 base score, which is reduced by a penalty coefficient for every infraction.

$$\text{IS} = \prod_{j}^{\text{Ped,Veh,Stat,Red}} (p^j)^{\#\text{ infractions}^j} \tag{7}$$

The penalty coefficient for each infraction is pre-defined and set to 0.50 for collision with a pedestrian, 0.60 for collision with a vehicle, 0.65 for collision with static layout, and 0.7 for red light violations. The official CARLA leaderboard also mentions a penalty for stop sign violations. However, we observe that none of our submissions have any stop sign infractions. Hence, we omit this infraction from our analysis.

(3) **Driving Score (DS)**: weighted average of the route completion with infraction multiplier P_i

$$\text{DS} = \frac{1}{N}\sum_{i}^{N} R_i P_i \tag{8}$$

(4) **Infractions per km**: the infractions considered are collisions with pedestrians, vehicles, and static elements, running a red light, off-road infractions, route deviations, timeouts, and vehicle blocked. We report the total number of infractions, normalized by the total number of km driven.

$$\text{Infractions per km} = \frac{\sum_i^N \#\text{ infractions}_i}{\sum_i^N k_i} \tag{9}$$

where k_i is the driven distance (in km) for route i. The Off-road infraction is slightly different. Instead of the total *number* of infractions the sum of *km driven off-road* is used. We multiply by 100 because this metric is a percentage.

4.6 Baselines

We compare our TransFuser model to several baselines. (1) **WOR** [110]: this is a multi-stage training approach that supervises the driving task with a Q function obtained using dynamic programming. It is the current state-of-the-art approach on the simpler NoCrash benchmark [38] for CARLA version 0.9.10. We use the author-provided pretrained model for evaluating this approach. (2) **Latent TransFuser**: to investigate the importance of the LiDAR input, we implement an auto-regressive waypoint prediction network that has the same architecture as the TransFuser but uses a fixed positional encoding image as input instead of the BEV LiDAR, as described in Section 3.7. (3) **LAV** [46]: this is a concurrent approach that performs sensor fusion via PointPainting [47], which concatenates semantic class information extracted from the RGB image to the LiDAR point cloud, to train a privileged motion planner to predict trajectories of all nearby vehicles in the scene. This privileged planner is then distilled into a policy that drives from raw sensor inputs only. We use the checkpoint publicly released by the authors[3] for our experiments. We note that this published checkpoint is not the exact same model as the one used for LAV's leaderboard entry. (4) **Late Fusion**: we implement a version of our architecture where the image and the LiDAR features are extracted independently using the same encoders as TransFuser but without the transformers (similar to [35]). The features from each branch are then fused through element-wise summation and passed to the waypoint prediction network. (5) **Geometric Fusion**: we implement a multi-scale geometry-based fusion method, inspired by [20], [22], involving both image-to-LiDAR and LiDAR-to-image feature fusion. We unproject each 0.125m $\times$ 0.125m block in our LiDAR BEV representation into 3D space, resulting in a 3D cell. We randomly select 5 points from the LiDAR point cloud lying in this 3D cell and project them into the image space. Then, we aggregate the image features of these points via element-wise summation before passing them to a 3-layer MLP. The output of the MLP is then combined with the LiDAR BEV feature of the corresponding 0.125m $\times$ 0.125m block at multiple resolutions throughout the feature extractor. Similarly, for each image pixel, we aggregate information from the LiDAR BEV features at multiple resolutions. This baseline is equivalent to replacing the transformers in our architecture with projection-based feature fusion. We also report results for the expert used for generating our training data, which defines an upper bound for the performance on the Longest6 evaluation setting. We provide additional details regarding the baselines in the supplementary material.

2. https://leaderboard.carla.org/scenarios/

3. https://github.com/dotchen/LAV

Method	DS ↑	RC ↑	IS ↑	Ped ↓	Veh ↓	Stat ↓	Red ↓	OR ↓	Dev ↓	TO ↓	Block ↓
WOR [110]	20.53 ± 3.12	48.47 ± 3.86	**0.56** ± 0.03	0.18	1.05	0.37	1.28	0.47	0.88	0.08	0.20
Latent TransFuser (Ours)	37.31 ± 5.35	**95.18** ± 0.45	0.38 ± 0.05	**0.03**	3.66	0.18	0.13	**0.04**	**0.00**	0.12	**0.05**
LAV [46]	32.74 ± 1.45	70.36 ± 3.14	0.51 ± 0.02	0.16	**0.83**	0.15	0.96	0.42	0.06	0.12	0.45
Late Fusion (LF)	22.47 ± 3.71	83.30 ± 3.04	0.27 ± 0.04	0.05	4.63	0.28	**0.11**	0.48	0.02	0.11	0.21
Geometric Fusion (GF)	27.32 ± 0.80	91.13 ± 0.95	0.30 ± 0.01	0.06	4.64	0.17	0.13	0.48	**0.00**	**0.05**	0.11
TransFuser (Ours)	**47.30** ± 5.72	93.38 ± 1.20	0.50 ± 0.06	**0.03**	2.45	**0.07**	0.16	**0.04**	**0.00**	0.06	0.10
Expert	*76.91 ± 2.23*	*88.67 ± 0.56*	*0.86 ± 0.03*	*0.02*	*0.28*	*0.01*	*0.03*	*0.00*	*0.00*	*0.08*	*0.13*

TABLE 1: **Longest6 Benchmark Results.** We compare our TransFuser model with several baselines in terms of driving performance and infractions incurred. We report the metrics for 3 evaluation runs of each model on the Longest6 evaluation setting. For the primary metrics (DS: Driving Score, RC: Route Completion, IS: Infraction Score) we show the mean and std. For the remaining infractions per km metrics (Ped: Collisions with pedestrians, Veh: Collisions with vehicles, Stat: Collisions with static layout, Red: Red light violation, OR: Off-road driving, Dev: Route deviation, TO: Timeout, Block: Vehicle Blocked) we show only the mean. TransFuser obtains the best DS by a large margin.

4.7 Implementation Details

We use 2 sensor modalities, the front-facing cameras and LiDAR point cloud converted to a BEV representation (Section 3.2), i.e., $S = 2$. The camera inputs are concatenated to a single image and encoded using a RegNetY-3.2GF [113] which is pretrained on ImageNet [114]. We use pre-trained models from [115]. The LiDAR BEV representation is encoded using another RegNetY-3.2GF [113] which is trained from scratch. Similar to [87], we perform angular viewpoint augmentation for the training data, by randomly rotating the sensor inputs by $\pm 20°$ and adjusting the waypoint labels according to this rotation. We use 1 transformer per resolution and 4 attention heads for each transformer. We select D_q, D_k, D_v from $\{72, 216, 576, 1512\}$ for the 4 transformers corresponding to the feature embedding dimension D_f at each resolution. We train the models with 4 RTX 2080Ti GPUs for 41 epochs, with an initial learning rate of 10^{-4} and a batch size of 12. We reduce the learning rate by a factor of 10 after epoch 30 and 40. We evaluate the epochs 31, 33, 35, 37, 39 and 41 closed loop on the validation routes of [39] for one seed and pick the epoch with the highest driving score. For all models, we use the AdamW optimizer [116], which is a variant of Adam. Weight decay is set to 0.01, and Adam beta values to the PyTorch defaults of 0.9 and 0.999.

4.8 Longest6 Benchmark Results

We begin with an analysis of driving performance on CARLA on the new Longest6 evaluation setting (Table 1). For each experimental result, the evaluation is repeated 3 times to account for the non-determinism of the CARLA simulator. Furthermore, imitation based methods typically show variation in performance when there is a change in the random initialization and data sampling due to the training seed [41], [43]. To account for the variance observed between different training runs, we use 3 different random seeds for each method, and report the metrics for an ensemble of these 3 training runs.

Latent TransFuser as a Strong Baseline: In our first experiment, we examine the performance of image-based methods. From the results in Table 1, we observe that WOR performs poorly on the Longest6 evaluation setting. In particular, we observe that WOR suffers from a poor RC with a much larger number of route deviations (Dev) than the remaining methods. On the other hand, we find that Latent Transfuser obtains the best RC in Table 1, with zero route deviations. This is likely because Latent TransFuser uses our inverse dynamics model (PID controller) for low-level control and represents goal locations in the same BEV coordinate space in which waypoints are predicted. In contrast, WOR uses coarse navigational commands (e.g. follow lane, turn left/right, change lane) to inform the model regarding driver intentions, and chooses an output action from a discrete action space. This result indicates that the TransFuser architecture involving auto-regressive waypoint prediction a strong baseline for the end-to-end driving task, even in the absence of a LiDAR sensor.

Sensor Fusion Methods: The goal of this experiment is to determine the impact of the LiDAR modality on the driving performance and compare different fusion methods. For this, we compare TransFuser to three baselines, LAV, Late Fusion (LF), Geometric Fusion (GF). LAV performs worse than TransFuser in terms of DS. The main difference arises from the 23% lower RC. Potential reasons could be worse steering as indicated by the higher off-road infractions, or false positives in the modular components which might be the reason for the higher blocked infraction. While LAV obtains a similar IS to TransFuser, upon probing further, we notice that it is only better in terms of avoiding collisions with vehicles and TransFuser performs better with respect to all other infractions. We note that in the Longest6 benchmark, there are a few routes where the vehicle is required to drive in dense traffic on multi-lane highways. TransFuser fails at lane merging in these situations, incurring a large amount of vehicle collisions (>20), strongly affecting its vehicle collision metric. Surprisingly, we observe that LF and GF perform worse than the image-only Latent TransFuser baseline (Table 1). The multi-scale geometry-based fusion encoder of GF gives some improvements when compared to LF, however, both LF and GF suffer from a poor IS. We hypothesize that this occurs because they do not incorporate global contextual reasoning which is necessary to safely navigate the intersections, and focus primarily on navigation to the goal at all costs while ignoring obstacles, which leads to several infractions. In contrast, our TransFuser model obtains an absolute improvement 19.98% in terms of DS when compared to GF. It also achieves an 48.59% reduction compared to LF and 47.64% reduction compared to GF in collisions per kilometer (Ped+Veh+Stat), and an absolute improvement of over 0.2 in its IS. These results

Fig. 5: **Lane Change Failures.** TransFuser fails at lane changes in dense traffic incurring a high number of consecutive collisions in routes where these situations occur. Two examples are shown in the top and bottom rows. Time goes forward from left to right.

Method	Single Model	Ensemble (3)
Late Fusion (LF)	23.5	46.7
Geometric Fusion (GF)	43.5	69.1
TransFuser (Ours)	27.6	59.6

TABLE 2: **Runtime.** We show the runtime per frame in ms for each method averaged over all timesteps in a single evaluation route. We measure runtimes for both a single model and an ensemble of three models. A single TransFuser model runs in real-time on an RTX 3090 GPU.

Method	LiDAR?	DS ↑	RC ↑	IS ↑
NEAT [39]	-	21.83	41.71	0.65
MaRLn [109]	-	24.98	46.97	0.52
WOR [110]	-	31.37	57.65	0.56
GRIAD [117]	-	36.79	61.86	0.60
Latent TransFuser (Ours)	-	45.20	66.31	**0.72**
Late Fusion (LF)	✓	26.07	64.67	0.47
Geometric Fusion (GF)	✓	41.70	87.85	0.47
TransFuser (Ours)	✓	61.18	86.69	0.71
LAV* [46]	✓	**61.85**	**94.46**	0.64

TABLE 3: **CARLA Leaderboard Evaluation.** We report the DS, RC, and IS over the 100 secret routes of the official evaluation server. Latent TransFuser and TransFuser improve the IS by a large margin in comparison to existing methods. *The LAV leaderboard entry uses an updated model different from the public checkpoint in Table 1.

indicate that attention is effective in incorporating the global context of the 3D scene, which allows for safer driving.

Limitations: We observe that all methods with a high RC struggle with vehicle collisions (Veh). Avoiding collisions is very challenging in our evaluation setting due to the traffic density being set to the maximum allowed density in CARLA. In particular, TransFuser has around 9× more vehicle collisions per kilometer than the expert. We observe that these collisions primarily occur during unprotected turns and lane changes as is illustrated in Fig. 5.

Runtime: We measure the runtime of each method on a single RTX 3090 GPU by averaging over all time-steps of one evaluation route. The runtime considered includes sensor data pre-processing, model inference and PID control. The results are shown in Table 2. We observe that the transformers in our architecture increase the runtime relative to the LF baseline by 17% for a single model and 28% for an ensemble of three models. However, a single TransFuser model can still be executed in real-time on this hardware. The GF baseline is slower than TransFuser despite its simpler fusion mechanism due to the extra time taken finding correspondences between the image and LiDAR tokens.

4.9 Leaderboard Results

We submit the models from our study to the CARLA autonomous driving leaderboard which contains a secret set of 100 evaluation routes and report the results in Table 3. Among the models that do not use LiDAR inputs, Latent TransFuser achieves the best performance. It obtains a DS of 45.20, which is nearly 10 points better than the next best image-based method, GRIAD [117]. GRIAD builds on top of the Reinforcement Learning (RL) method presented in [109]. In this approach, an encoder is first trained to predict both the 2D semantics and specific affordances such as the scene traffic light state, and the relative position and orientation between the vehicle and lane. The encoder is then frozen and used to train a value function-based RL method using a replay buffer that is partially filled with expert demonstrations. GRIAD requires 45M samples from the CARLA simulator for training [117] whereas our training dataset has only 228k frames (200× less than GRIAD).

For the LiDAR-based baselines, GF performs better than LF, similar to our findings on Longest6. Incorporating global attention via TransFuser leads to further improvements with a state-of-the-art IS. While LAV performs similarly to TransFuser, it is only marginally better in terms of DS (+0.67), which is likely within the evaluation variance. To obtain this marginal improvement, LAV adopts multi-stage training with several pretrained modular components and a teacher-student distillation framework. In contrast, TransFuser achieves state-of-the-art results with a straightforward single-stage IL training procedure. For further improvements, the training procedure of TransFuser can potentially be combined with techniques used in previous work on autonomous driving such as Active Learning [118]–[120],

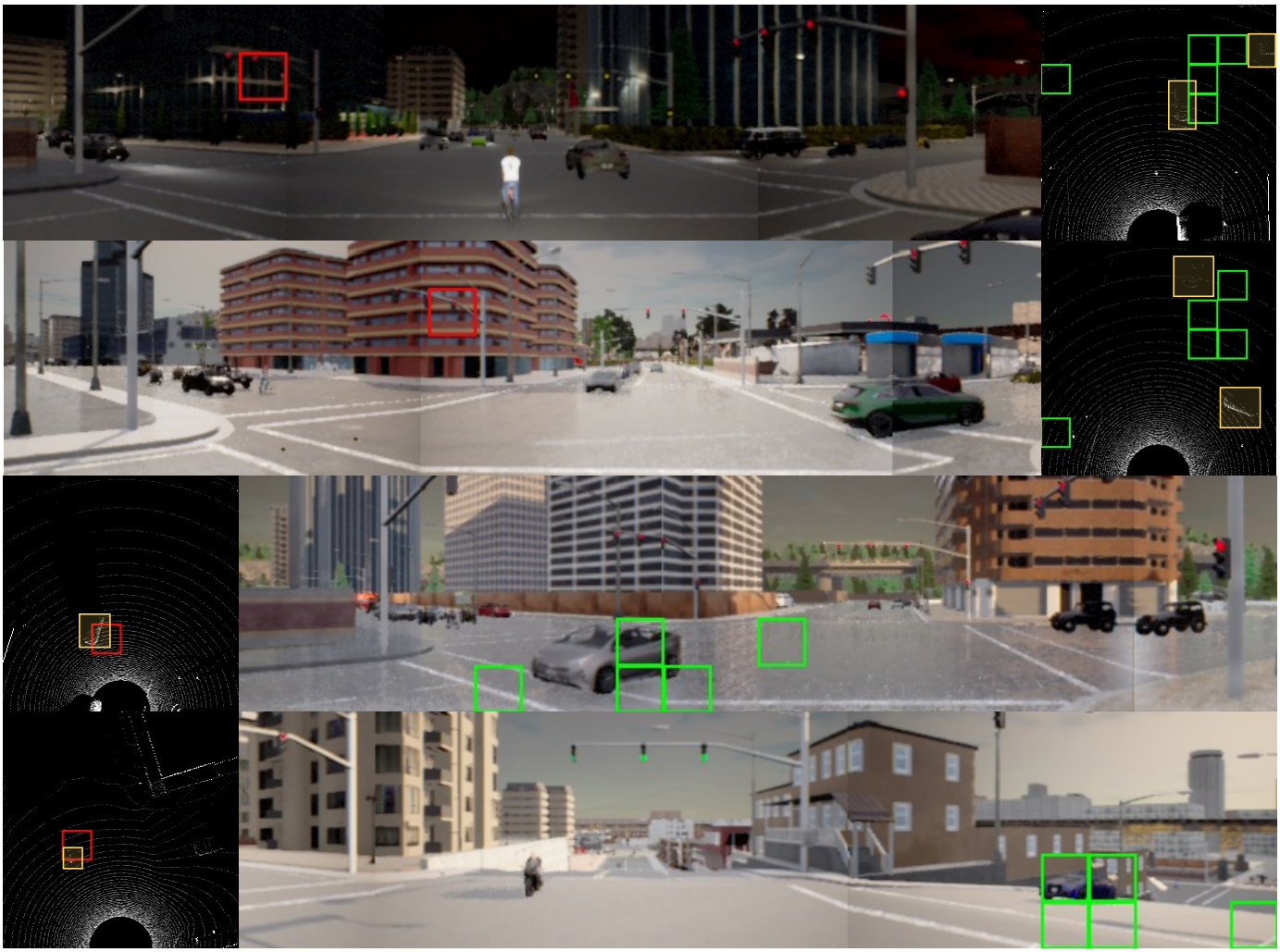


Fig. 6: **Attention Maps.** For the red query token, we show the top-5 attended tokens in green and highlight the presence of vehicles in the LiDAR point cloud in yellow. Top 2 rows: image to LiDAR, bottom 2 rows: LiDAR to image. TransFuser attends to areas near vehicles and traffic lights at intersections.

Head	T1		T2		T3		T4	
	I_t	L_t	I_t	L_t	I_t	L_t	I_t	L_t
1	100.00	0.00	99.83	0.00	44.55	98.69	77.79	89.97
2	100.00	0.00	99.99	0.00	07.58	98.98	80.05	95.91
3	100.00	0.00	39.71	0.00	27.73	98.09	90.08	99.98
4	99.99	1.45	99.99	0.26	99.99	99.98	80.13	99.47

TABLE 4: **Cross-Modal Attention Statistics.** We report the % of tokens (I_t: Image tokens, L_t: LiDAR tokens) for which at least 1 of the top-5 attended tokens belongs to the other modality for each head of the four transformers: T1, T2, T3, T4.

DAgger [121], [122], adversarial simulation [123]–[125], RL-based fine-tuning [100] and teacher-student distillation [46], [87], [110], [126].

4.10 Attention Statistics and Visualizations

Our architecture (Fig. 2) consists of 4 transformers with 4 attention layers and 4 attention heads in each transformer. In this section, we visualize the attention maps from the final attention layer for each head for each transformer. The transformer takes in 110 image feature tokens and 64 LiDAR feature tokens as input where each token corresponds to a 32×32 patch in the input modality. We consider intersection scenarios from Town03 and Town05 and examine the top-5 attention weights for the 66 tokens in the 2^{nd}, 3^{rd} and 4^{th} rows of the image feature map and the 24 tokens in the 4^{th}, 5^{th} and 6^{th} rows of the LiDAR feature map. We select these tokens since they correspond to the intersection region in the input modality and contain traffic lights and vehicles.

We compute statistics on cross-modal attention for image and LiDAR feature tokens. Specifically, we report the % of tokens for which at least one of the top-5 attended tokens belong to the other modality for each head of the 4 transformers (T1, T2, T3, T4) in Table 4. We observe that the earlier transformers have negligible LiDAR to image attention compared to later transformers in which nearly all the LiDAR tokens aggregate information from the image features. Furthermore, different heads of each transformer also show distinctive behavior, e.g. head 3 of T2 has significantly less cross-attention for image tokens than other heads, head 2 of T3 has very little cross-attention whereas head 4 has significantly higher cross-attention for image

Method	Ensemble?	Safety Heuristic	Longest6			Leaderboard		
			DS ↑	RC ↑	IS ↑	DS ↑	RC ↑	IS ↑
Latent TransFuser	-	Global	**50.00 ± 1.13**	90.38 ± 3.32	**0.56 ± 0.02**	47.05	71.66	0.72
	-	Creep Only	42.19 ± 5.49	94.84 ± 1.40	0.44 ± 0.06	42.36	86.67	0.51
	✓	Creep Only	36.18 ± 5.36	**95.34 ± 2.20**	0.37 ± 0.05	45.03	75.37	0.62
TransFuser	-	Global	49.49 ± 8.63	90.67 ± 4.78	0.55 ± 0.09	41.93	58.55	**0.77**
	-	Creep Only	42.51 ± 2.49	91.01 ± 0.83	0.46 ± 0.02	50.57	73.84	0.68
	✓	Creep Only	47.30 ± 5.72	93.38 ± 1.20	0.50 ± 0.06	**61.18**	**86.69**	0.71

TABLE 5: **Impact of Global Safety Heuristic.** The heuristic leads to consistent minor improvements for Latent TransFuser. For TransFuser, though the heuristic improves the Longest6 scores, it reduces the performance on the CARLA leaderboard.

tokens compared to other heads. Overall, T4 exhibits extensive cross-attention for both image and LiDAR tokens, which indicates that TransFuser is effective in aggregating information between the two modalities.

We show four cross-attention visualizations for T4 in Fig. 6. We observe a common trend in attention maps: TransFuser attends to areas near vehicles and traffic lights at intersections, albeit at a slightly different location in the image and LiDAR feature maps. Additional visualizations for all the transformers are provided in the supplementary.

4.11 Global Safety Heuristic

The primary motivation of the safety heuristic described in Section 3.5 is to prevent collisions during the applied creeping behavior. Therefore, in Tables 1 and 3, we apply the safety heuristic during creeping only. However, rule-based fallback systems have been shown to improve the safety of IL models [99]. In this experiment, we investigate the impact of applying the safety heuristic described in Section 3.5 globally, i.e. during both creeping and regular driving. We show results on both the Longest6 benchmark and the CARLA leaderboard. To reduce the computational overhead of this analysis, we evaluate a single model instead of an ensemble of 3 different training runs, which were used in Tables 1 and 3. However, for clarity, we also report the scores of the ensemble. Additionally, the results shown for Latent TransFuser are from preliminary experiments where we include a LiDAR sensor and use the LiDAR-based safety heuristic instead of the CenterNet based intersection check described in Section 3.7, leading to minor differences when compared to the numbers from Tables 1 and 3.

The results are shown in Table 5. For Latent TransFuser, we observe that the global safety heuristic improves the DS by 8 points on the Longest6 benchmark and 5 points on the leaderboard compared to it being applied during creeping only. In particular, this is due to a large improvement in the IS (e.g. from 0.51 to 0.72 on the leaderboard). Interestingly, we observe a different trend for TransFuser. The global safety heuristic improves the DS by 7 points on the Longest6 benchmark where we tuned the hyper-parameters (i.e., size of the rectangular safety box). However, it leads to a drop of nearly 10 points in DS on the leaderboard. The global safety heuristic leads to reduced route completion for both methods on the CARLA leaderboard. This indicates that the heuristic works well on observed maps, but does not generalize to unknown and potentially unseen road layouts. For TransFuser, which already had a higher infraction score than Latent TransFuser, the improvement in IS does not make up for the reduction in RC leading to an overall reduction in DS through the safety heuristic.

When comparing the results of a single model and the corresponding ensemble, we find that ensembling improves the DS for both Latent TransFuser and TransFuser on the leaderboard, in particular for TransFuser which improves by more than 10 points. On the Longest6 routes, the ensembling has a lower impact of 5 points for the TransFuser and even a reduction in performance for Latent TransFuser. The single models reported in Table 5 are the first of three training seeds (Table 6). Ensembling might have a larger positive impact on TransFuser because the models had a larger training variance, which we discuss in the following.

Training	Eval	LTF	LF	GF	TF
1	1	48.82	31.94	46.13	59.45
	2	50.12	33.29	43.62	44.15
	3	51.07	34.79	39.42	44.87
	avg.	50.00	33.34	43.06	49.49
2	1	44.53	37.05	36.64	51.50
	2	54.35	36.79	39.40	50.91
	3	52.57	47.98	34.93	52.35
	avg.	50.48	40.60	36.99	51.59
3	1	51.15	34.90	49.77	59.76
	2	48.10	32.47	45.64	57.39
	3	49.80	44.98	52.47	52.88
	avg.	49.68	37.45	49.30	56.68

TABLE 6: **Training and evaluation variance.** We show the DS of each evaluation on the Longest6 benchmark. We train each baseline 3 times, and perform 3 evaluation runs of each individual trained model. LTF: Latent TransFuser, LF: Late Fusion, GF: Geometric Fusion, TF: TransFuser. All models exhibit large variance in scores.

Training Seed Variance: We show the impact of training and evaluation seed variance in Table 6. We train each baseline 3 times and evaluate each model 3 times on the Longest6 routes with the global safety heuristic enabled. We observe that the best achieved score can differ from the worst score by 10-15 points for an individual model, leading to extremely large variance. In particular, for the first seed of TransFuser, the DS ranges from 44.15 to 59.45. For Geometric Fusion, the average DS differs by 12 points between the worst and best training seed. This amount of variance is problematic when trying to analyze or compare different approaches. We would like to emphasize that the variance reported in Table 6 comes from two factors. The training variance between different seeds results from different network initializations, data sampling and data augmentations during optimization. The evaluation variance is a result

Auxiliary Losses	DS ↑	RC ↑	IS ↑
None	44.29	78.17	0.58
No Depth	56.23	91.80	0.61
No Semantics	53.76	88.40	0.61
No HD Map	50.96	89.52	0.58
No Vehicle Detection	53.43	88.49	0.60
All Losses (Worst Seed)	49.49	90.67	0.55
All Losses (Best Seed)	**56.68**	**92.28**	**0.62**

TABLE 7: **Auxiliary Tasks.** The results shown are the mean over 3 evaluations on Longest6. Training without auxiliary losses leads to a significant reduction in RC and DS.

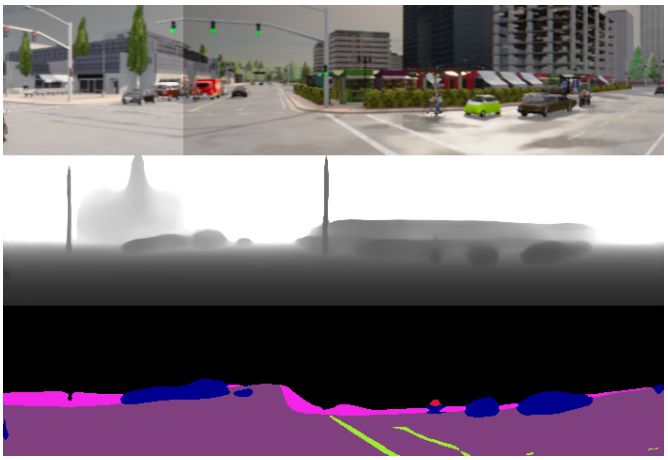
of the variation in the traffic manager, physics and sensor noise of CARLA 0.9.10. Based on the results observed, the randomness in evaluation is the primary cause of variance, in addition to secondary training seed variance, but both factors are considerable. The existing practice for state-of-the-art methods on CARLA is to report only the evaluation variance by running multiple evaluations of a single training seed. This may lead to premature conclusions (e.g. when considering only the three evaluations of the first training seed, Latent TransFuser outperforms TransFuser). We argue (given these findings) that future studies should report results by varying the training seed for both the baselines and proposed new methods, in addition to the results of the best seed or ensemble.

4.12 Ablation Studies

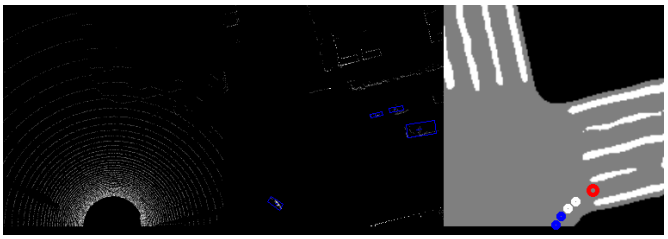
We now analyze several design choices for TransFuser in a series of ablation studies on the Longest6 benchmark. Since the global safety heuristic leads to consistent and significant improvements for TransFuser on the Longest6 routes (Table 5), we use this setting for the ablation studies. The evaluation is repeated 3 times for each experiment, however, we use a single training run for these results instead of an ensemble of 3 different training runs as in Table 1. To further reduce the computational overhead, we always evaluate epoch 31, as we have observed in preliminary experiments that it is usually close to the best epoch in performance. For the default configuration, we have 3 available training runs. We report the best and worst training seed to account for the randomness due to training. Ablations lying within this interval likely do not have a large impact.

Auxiliary Tasks: As described in Section 3.6, we consider 4 auxiliary tasks in this work. In Table 7 we show the performance of TransFuser when all these auxiliary losses are removed, as well as the impact of removing each loss independently. We observe that with no auxiliary tasks, there is a steep drop in RC from 92.28 to 78.17. Removing only a single auxiliary task does not have a large impact. All results lie within the range between the best and worst seed of the default configuration in terms of driving score. In Fig. 7, we visualize the predictions made by TransFuser when trained with all 4 auxiliary losses.

Architecture: In Table 8, we analyze the impact of varying the TransFuser encoder architecture. We study the importance of fusion in both directions by selectively removing the residual output connections from the fusion transformers to the convolutional backbones. Fusion for only the



(a) Top to bottom: image input, predicted depth, predicted semantics (legend: none, road, lane, sidewalk, vehicle, person).



(b) Left to right: LiDAR ground plane channel, bounding box predictions overlaid on LiDAR obstacle channel (points above ground plane), HD map prediction.

Fig. 7: **Visualization of Auxiliary Tasks.** We visualize the inputs and outputs of both the image branch and LiDAR branch for the same driving scene. Further, the input target point is visualized as a red circle and predicted waypoints as blue and white circles on the HD map prediction. The first two waypoints (which are used to obtain the steering angle for our PID controller) are shown as blue circles, and the remaining two waypoints as white circles.

Camera → LiDAR or only the LiDAR → Camera direction gives a slightly lower performance that the default model with bi-directional fusion. Removing the fusion mechanism in the early blocks of the encoder and performing feature fusion at only the deepest 1, 2 and 3 scales also leads to a small drop in performance. For the fusion transformers, we find that 2-8 attention layers give similar performance. The default resolution of the image features is 22×5 and the LiDAR features is 8×8. We observe that using each of these 22×5 + 8×8 features as independent input tokens for the transformer leads to better results when compared to a fusing information across different image and LiDAR resolutions through a reduced image token count of 11×3 or LiDAR token count of 4×4. We also evaluate a version of TransFuser where the input to the MLP and GRU decoder from the final fusion transformer is obtained via a dedicated attention token instead of the default global average pooling. This is a standard idea for attention-based averaging of spatial features, similar to the CLS token of Vision Transformers [37]. We find that the default architecture with global average pooling is simpler to implement and performs similarly in practice. The most impactful

Parameter	Value	DS ↑	RC ↑	IS ↑
Fusion Direction	LiDAR → Camera	46.36	87.46	0.55
	Camera → LiDAR	47.99	86.24	0.57
Fusion Scales	1	49.35	84.47	0.57
	2	53.52	91.78	0.59
	3	48.77	85.33	0.60
Attention Layers	2	53.49	90.65	0.60
	6	56.24	92.56	0.61
	8	56.13	**92.61**	0.61
Token Count	11×3 + 8×8	45.63	90.32	0.51
	22×5 + 4×4	49.14	87.10	0.56
Averaging	Attention Token	54.21	90.78	0.60
Backbones	Res34-Res18	42.01	92.43	0.45
	Reg0.8-Reg0.8	44.38	87.96	0.50
	Reg1.6-Reg1.6	47.80	91.62	0.52
	NeXt-T-NeXt-T	48.27	92.30	0.53
Default Config	Worst Seed	49.49	90.67	0.55
	Best Seed	**56.68**	92.28	**0.62**

TABLE 8: **Architecture Ablations.** The results shown are the mean over 3 evaluations on Longest6. The default configuration fuses in both directions. It uses 4 fusion scales, 4 attention layers, 22×5 + 8×8 tokens, global average pooling, and RegNetY-3.2GF backbones. The encoder backbone has the highest impact on the final driving score.

Parameter	Value	DS ↑	RC ↑	IS ↑
LiDAR Range	64m × 32m	49.08	91.10	0.54
	64m × 64m	47.55	90.72	0.52
LiDAR Encoder	PointPillars	50.83	91.56	0.55
Camera FOV	120°	49.90	90.05	0.56
	90°	42.18	88.49	0.51
No Rasterized Goal	-	54.80	91.63	0.60
No Rotation Aug	-	**56.85**	**92.73**	0.61
Default Config	Worst Seed	49.49	90.67	0.55
	Best Seed	56.68	92.28	**0.62**

TABLE 9: **Model Input Ablations.** The results shown are the mean over 3 evaluations on Longest6. The default configuration uses a 32m × 32m LiDAR range and 132° camera FOV. Camera FOV has the largest impact on the DS.

Velocity Input?	Creeping?	DS ↑	RC ↑	IS ↑
-	-	46.35	78.28	0.63
	✓	**56.68**	**92.28**	0.62
✓	-	37.34	64.27	**0.65**
	✓	45.35	86.22	0.52

TABLE 10: **Inertia Problem.** The results shown are the mean over 3 evaluations on Longest6. Creeping improves the RC in both the setting where we input the velocity to our encoder and our default configuration (no velocity input).

architecture choice is the backbone architecture for both branches. The default configuration of RegNetY-3.2GF backbones outperforms ConvNeXt-Tiny [127], RegNetY-1.6GF and RegNetY-0.8GF based backbones. We also observe a large improvement over the use of a ResNet34 for the image branch and ResNet18 for the BEV branch, as in [41], which leads to a model with lower network capacity.

Model Inputs: As shown in Table 9, increasing the LiDAR range or reducing the camera FOV from the default configuration leads to a reduced IS and a corresponding drop in DS. Our method works with arbitrary grids as inputs. Therefore, it could benefit from orthogonal improvements in LiDAR encoding. However, we did not observe improvements by using a learned LiDAR encoder [128], and hence stick with the simpler voxelization approach. This model was trained with batch size 10 due to its higher memory requirements. Interestingly, despite being useful in our preliminary experiments, removing the rasterized goal location channel from the LiDAR branch, or removing the random rotation of sensor inputs by ±20° used during data augmentation show only a small impact on the performance in the final configuration which is unlikely to be significant.

Inertia Problem: As we note in Section 3.5, we add creeping to our controller to prevent the agent from being overly cautious. This type of behavior, called the inertia problem [38] is typically attributed to the spurious correlation that exists between input velocity and output acceleration in an IL dataset. Interestingly, though we do not use velocity as an input to our models, we observe that creeping in the controller increases the RC significantly while maintaining a similar IS (Table 10). This indicates that a factor besides the velocity input, such as an imbalance in training data distribution, may be a key contributing factor to the inertia problem. We also train a version of TransFuser where we provide the current velocity as input by projecting the scalar value into the same dimensions as the transformer positional embedding using a linear layer. This velocity embedding is combined with the learnable positional embedding through element-wise summation and fed into the transformer at all 4 stages of the backbone. Including the velocity input leads to a sharp drop in DS, which cannot be recovered through the creeping behavior.

5 Discussion and Conclusions

In this work, we demonstrate that IL policies based on existing sensor fusion methods suffer from high infraction rates in complex driving scenarios. To overcome this limitation, we present a novel multi-modal fusion transformer (TransFuser) for integrating representations of different modalities. TransFuser uses attention to capture the global 3D scene context and focuses on dynamic agents and traffic lights, resulting in state-of-the-art performance on CARLA with a significant reduction in infractions. Given that our method is simple, flexible and generic, it would be interesting to explore it further with additional sensors, e.g. radar, or apply it to other embodied AI tasks.

We hope that the proposed benchmark with long routes and dense traffic will become a suitable option for the community to conduct ablation studies or obtain detailed statistics that are not feasible via the CARLA leaderboard.

Our study has several limitations. We have provided a simple solution to the inertia problem (creeping), but this deserves more study. Due to the sensor limits of the CARLA leaderboard, our sensor setup does not generate data from the rear of the vehicle, which is relevant in lane change situations. We only investigate single time step input data in this work. Processing temporal inputs is likely necessary to reduce vehicle collisions in intersections by enabling estimation of the acceleration and velocity of other traffic participants. We do not investigate the impact of latency

on the final driving performance, which has been shown to be important for real-world applications [129]. This is because the default configuration of the CARLA simulator waits for the agent to finish its computation before it resumes simulation of the world. Finally, all our experiments are only conducted in simulation. Real-world data is more diverse and can have more challenging noise. We make use of multiple high-quality labels that the CARLA simulator provides, such as dense depth maps. Real-world datasets often do not provide labels of such high quality and might not provide all the types of labels we have used in this work.

Progress on the CARLA leaderboard has been rapid, with the state-of-the-art scores increasing from the range of 20 to 60 in the short time period since the preliminary version of this work at CVPR 2021. As novel submissions to the leaderboard move towards alternatives to end-to-end IL that involve complex multi-stage training or RL-based objectives, we show that a simple IL training procedure with our proposed architecture is highly competitive. Future works should consider our Latent Transfuser as a standard baseline for image-only IL. Based on our analysis, we believe that overcoming the inertia problem in a principled manner and reducing both training and evaluation variance will be key challenges for IL-based driving.

Acknowledgements

This work was supported by the BMWi in the project KI Delta Learning (project number: 19A19013O) and the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B. Andreas Geiger was supported by the ERC Starting Grant LEGO-3D (850533) and the DFG EXC number 2064/1 - project number 390727645. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Kashyap Chitta, Bernhard Jaeger and Katrin Renz. The authors also thank Pavan Teja Varigonda for his help with sampling the training routes and Niklas Hanselmann for proofreading.

References

[1] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control," in *Proc. of the International Conf. on Learning Representations (ICLR)*, 2020. 1, 3

[2] A. Filos, P. Tigas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" in *Proc. of the International Conf. on Machine learning (ICML)*, 2020. 1, 3, 5

[3] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1, 5

[4] A. Cui, A. Sadat, S. Casas, R. Liao, and R. Urtasun, "Lookout: Diverse multi-future prediction and planning for self-driving," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 1

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. on Robot Learning (CoRL)*, 2017. 1, 2, 6

[6] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1

[7] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[8] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1

[11] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[12] "Waymo open dataset: An autonomous driving dataset," https://www.waymo.com/open, 2019. 1

[13] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling," *arXiv.org*, vol. 1805.04687, 2018. 1

[14] S. Fadadu, S. Pandey, D. Hegde, Y. Shi, F. Chou, N. Djuric, and C. Vallespi-Gonzalez, "Multi-view fusion of sensor data for improved perception and prediction in autonomous driving," *arXiv.org*, vol. 2008.11901, 2020. 1, 2

[15] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[16] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," in *Proc. Conf. on Robot Learning (CoRL)*, 2019. 1, 2

[17] K. Chen, R. Oldja, N. Smolyanskiy, S. Birchfield, A. Popov, D. Wehr, I. Eden, and J. Pehserl, "Mvlidarnet: Real-time multi-class scene understanding for autonomous driving using multiple views," *arXiv.org*, vol. 2006.05518, 2020. 1

[18] X. Qi, Q. Chen, J. Jia, and V. Koltun, "Semi-parametric image synthesis," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[19] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2018. 1, 2

[20] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2, 8

[21] Y. You, Y. Wang, W. Chao, D. Garg, G. Pleiss, B. Hariharan, M. E. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," in *Proc. of the International Conf. on Learning Representations (ICLR)*, 2020. 1, 2

[22] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 8

[23] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. 1, 2

[24] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[25] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data," in *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2020. 1, 2

[26] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun, "Pnpnet: End-to-end perception and prediction with tracking in the loop," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[27] Z. Zhang, J. Gao, J. Mao, Y. Liu, D. Anguelov, and C. Li, "Stinet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[28] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *Proc. Conf. on Robot Learning (CoRL)*, 2018. 1, 2

[29] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu, A. Dayan, S. Zhang, B. C. Becker, G. P. Meyer, C. Vallespi-Gonzalez, and C. K. Wellington, "Multixnet: Multiclass multistage multimodal motion prediction," *arXiv.org*, vol. 2006.02000, 2020. 1, 2

[30] G. P. Meyer, J. Charland, S. Pandey, A. Laddha, C. Vallespi-Gonzalez, and C. K. Wellington, "Laserflow: Efficient and probabilistic object detection and motion forecasting," *arXiv.org*, vol. 2003.05982, 2020. 1, 2

[31] L. L. Li, B. Yang, M. Liang, W. Zeng, M. Ren, S. Segal, and R. Urtasun, "End-to-end contextual perception and prediction with interaction transformer," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2020. 1, 2, 3

[32] C. Fu, C. Dong, C. Mertz, and J. M. Dolan, "Depth completion via inductive fusion of planar LIDAR and monocular camera," *arXiv.org*, vol. 2009.01875, 2020. 1

[33] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse lidar data with depth-normal constraints," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1

[34] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[35] I. Sobh, L. Amin, S. Abdelkarim, K. Elmadawy, M. Saeed, O. Abdeltawab, M. Gamal, and A. E. Sallab, "End-to-end multi-modal sensors fusion system for urban automated driving," in *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2018. 1, 2, 8

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1, 3

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. 1, 13

[38] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2, 3, 5, 6, 8, 14

[39] K. Chitta, A. Prakash, and A. Geiger, "Neat: Neural attention fields for end-to-end autonomous driving," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 3, 5, 6, 8, 9, 10

[40] "Carla autonomous driving leaderboard," https://leaderboard.carla.org/, 2020. 2

[41] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 9, 14

[42] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, "Multimodal end-to-end autonomous driving," *arXiv.org*, vol. 1906.03199, 2019. 2

[43] A. Behl, K. Chitta, A. Prakash, E. Ohn-Bar, and A. Geiger, "Label efficient visual abstractions for autonomous driving," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2020. 2, 9

[44] B. Zhou, P. Krähenbühl, and V. Koltun, "Does computer vision matter for action?" *Science Robotics*, vol. 4, no. 30, 2019. 2

[45] O. Natan and J. Miura, "Fully end-to-end autonomous driving with semantic depth cloud mapping and multi-agent," *arXiv.org*, vol. 2204.05513, 2022. 2

[46] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 8, 9, 10, 11

[47] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 8

[48] C. Chen, L. Z. Fragonara, and A. Tsourdos, "Roifusion: 3d object detection from lidar and vision," *arXiv.org*, vol. 2009.04554, 2020. 2

[49] A. Laddha, S. Gautam, S. Palombo, S. Pandey, and C. Vallespi-Gonzalez, "Mvfusenet: Improving end-to-end object detection and motion forecasting through multi-view fusion of lidar data," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021. 2

[50] S. Gautam, G. P. Meyer, C. Vallespi-Gonzalez, and B. C. Becker, "Sdvtracker: Real-time multi-sensor association and tracking for self-driving vehicles," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*, 2021. 2

[51] D. Dai, Z. Chen, P. Bao, and J. Wang, "A review of 3d object detection for autonomous driving of electric vehicles," *World Electric Vehicle Journal*, vol. 12, 2021. 2

[52] A. Mohta, F. Chou, B. C. Becker, C. Vallespi-Gonzalez, and N. Djuric, "Investigating the effect of sensor modalities in multi-sensor detection-prediction models," *arXiv.org*, vol. 2101.03279, 2021. 2

[53] O. Mees, A. Eitel, and W. Burgard, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2016. 2

[54] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[55] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from RGB-D data," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[56] S. Pang, D. D. Morris, and H. Radha, "Clocs: Camera-lidar object candidates fusion for 3d object detection," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2020. 2

[57] X. Zhao, Z. Liu, R. Hu, and K. Huang, "3d object detection using scale invariant and feature reweighting networks," in *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2019. 2

[58] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[59] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "FUTR3D: A unified sensor fusion framework for 3d detection," *arXiv.org*, vol. 2203.10642, 2022. 2

[60] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus, and S. Han, "BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation ," *arXiv.org*, vol. 2205.13542, 2022. 2

[61] Y. H. Khalil and H. T. Mouftah, "Licanext: Incorporating sequential range residuals for additional advancement in joint perception and motion prediction," *IEEE Access*, vol. 9, pp. 146 244–146 255, 2021. 2

[62] G. Wang, C. Peng, J. Zhang, and H. Wang, "Interactive multi-scale fusion of 2d and 3d features for multi-object tracking," *arXiv.org*, vol. 2203.16268, 2022. 2

[63] Z. Liu, T. Huang, B. Li, X. Chen, X. Wang, and X. Bai, "Epnet++: Cascade bi-directional fusion for multi-modal 3d object detection," *arXiv.org*, vol. 2112.11088, 2021. 2

[64] Y. Zhang, J. Chen, and D. Huang, "Cat-det: Contrastively augmented transformer for multi-modal 3d object detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[65] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2019. 2, 3

[66] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain inspired cognitive model with attention for self-driving cars," *arXiv.org*, vol. 1702.05596, 2017. 2

[67] Z. Yuan, X. Song, L. Bai, Z. Wang, and W. Ouyang, "Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021. 2

[68] H. Sheng, S. Cai, Y. Liu, B. Deng, J. Huang, X.-S. Hua, and M.-J. Zhao, "Improving 3d object detection with channel-wise transformer," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[69] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese, "Car-net: Clairvoyant attentive recurrent network," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 2

[70] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive GAN for predicting paths compliant to social and physical constraints," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[71] Y.-F. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[72] C. Choi and B. Dariush, "Looking to relations for future trajectory forecast," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[73] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. D. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[74] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[75] B. Wei, M. Ren, W. Zeng, M. Liang, B. Yang, and R. Urtasun, "Perceive, attend, and drive: Learning spatial attention for safe self-driving," *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2021. 2, 3

[76] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," *arXiv.org*, vol. 2103.14023, 2021. 2

[77] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens, "Scene transformer: A unified architecture for predicting multiple agent trajectories," *arXiv.org*, vol. 2106.08417, 2021. 2

[78] C. Gou, Y. Zhou, and D. Li, "Driver attention prediction based on convolution and transformers," *The Journal of Supercomputing*, 2022. 2

[79] Z. Cao, E. Biyik, G. Rosman, and D. Sadigh, "Leveraging smooth attention prior for multi-agent trajectory prediction," *arXiv.org*, vol. 2203.04421, 2022. 2

[80] W. Yang, Q. Li, W. Liu, Y. Yu, Y. Ma, S. He, and J. Pan, "Projecting your view attentively: Monocular road scene layout estimation via cross-view transformation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 6

[81] A. Saha, O. Mendez Maldonado, C. Russell, and R. Bowden, "Translating Images into Maps," in *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022. 3

[82] B. Zhou and P. Krähenbühl, "Cross-view Transformers for real-time Map-view Semantic Segmentation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[83] L. Peng, Z. Chen, Z. Fu, P. Liang, and E. Cheng, "BEVSegFormer: Bird's Eye View Semantic Segmentation From Arbitrary Camera Rigs," *arXiv.org*, vol. 2203.04050, 2022. 3

[84] E. Xie, Z. Yu, D. Zhou, J. Philion, A. Anandkumar, S. Fidler, P. Luo, and J. M. Alvarez, "M^2BEV: Multi-Camera Joint 3D Detection and Segmentation with Unified Birds-Eye View Representation," *arXiv.org*, vol. 2204.05088, 2022. 3

[85] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers," *arXiv.org*, vol. 2203.17270, 2022. 3

[86] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "PRECOG: prediction conditioned on goals in visual multi-agent settings," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 3

[87] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Proc. Conf. on Robot Learning (CoRL)*, 2019. 3, 5, 7, 9, 11

[88] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. LeCun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2005. 3

[89] D. Wang, C. Devin, Q. Cai, P. Krähenbühl, and T. Darrell, "Monocular plan view networks for autonomous driving," in *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2019. 3

[90] M. Bansal, A. Krizhevsky, and A. S. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proc. Robotics: Science and Systems (RSS)*, 2019. 3

[91] C. Wen, J. Lin, T. Darrell, D. Jayaraman, and Y. Gao, "Fighting copycat agents in behavioral cloning from observation histories," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[92] C. Wen, J. Lin, J. Qian, Y. Gao, and D. Jayaraman, "Keyframe-focused visual imitation learning," in *Proc. of the International Conf. on Machine learning (ICML)*, 2021. 3

[93] R. Bellman, *Adaptive Control Processes - A Guided Tour*. Princeton University Press, 2015, vol. 2045. 3

[94] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "Videobert: A joint model for video and language representation learning," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 4

[95] M. Chen, A. Radford, J. Wu, H. Jun, P. Dhariwal, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *Proc. of the International Conf. on Machine learning (ICML)*, 2020. 4

[96] D. Qi, L. Su, J. Song, E. Cui, T. Bharti, and A. Sacheti, "Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data," *arXiv.org*, vol. 2001.07966, 2020. 4

[97] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv.org*, vol. 2010.11929, 2020. 4

[98] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. 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. 5

[99] M. Vitelli, Y. Chang, Y. Ye, M. Wołczyk, B. Osiński, M. Niendorf, H. Grimmett, Q. Huang, A. Jain, and P. Ondruska, "Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies," *arXiv.org*, vol. 2109.13602, 2021. 5, 12

[100] E. Ohn-Bar, A. Prakash, A. Behl, K. Chitta, and A. Geiger, "Learning situational driving," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5, 11

[101] Z. Huang, C. Lv, Y. Xing, and J. Wu, "Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding," *IEEE Sensors Journal*, 2020. 5

[102] A. Loukkal, Y. Grandvalet, T. Drummond, and Y. Li, "Driving among Flatmobiles: Bird-Eye-View occupancy grids from a monocular camera for holistic trajectory planning," *arXiv.org*, vol. 2008.04047, 2020. 5

[103] A. Hu, F. Cotter, N. Mohan, C. Gurau, and A. Kendall, "Probabilistic future prediction for video scene understanding," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 5

[104] P. Li, X. Liang, D. Jia, and E. P. Xing, "Semantic-aware grad-gan for virtual-to-real urban scene adaption," *arXiv.org*, vol. 1801.01726, 2018. 5

[105] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv.org*, vol. 1904.07850, 2019. 6

[106] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 5632–5640. 6

[107] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[108] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Towards large-pose face frontalization in the wild," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 6

[109] M. Toromanoff, E. Wirbel, and F. 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. 6, 10

[110] D. Chen, V. Koltun, and P. Krähenbühl, "Learning to drive from a world on rails," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 6, 7, 8, 9, 10, 11

[111] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. 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. 7

[112] T. Agarwal, H. Arora, and J. Schneider, "Affordance-based reinforcement learning for urban driving," *arXiv.org*, vol. 2101.05970, 2021. 7

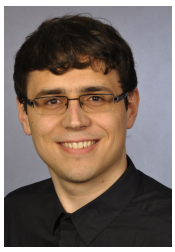
[113] I. Radosavovic, R. P. Kosaraju, R. B. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 9

[114] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, "Imagenet: A large-scale hierarchical image database," in *Proc.*

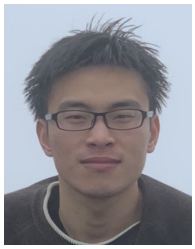
IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2009. 9
[115] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019. 9
[116] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019. 9
[117] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde, "GRI: general reinforced imitation and its application to vision-based autonomous driving," *arXiv.org*, vol. 2111.08575, 2021. 10
[118] K. Chitta, J. M. Alvarez, and A. Lesnikowski, "Large-scale visual active learning with deep probabilistic ensembles," *arXiv.org*, vol. 1811.03575, 2018. 10
[119] K. Chitta, J. M. Alvarez, E. Haussmann, and C. Farabet, "Training data subset search with ensemble active learning," *arXiv.org*, vol. 1905.12737, 2019. 10
[120] E. Haussmann, M. Fenzi, K. Chitta, J. Ivanecky, H. Xu, D. Roy, A. Mittel, N. Koumchatzky, C. Farabet, and J. M. Alvarez, "Scalable Active Learning for Object Detection," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2020. 10
[121] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 11
[122] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. 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. 11
[123] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "Advsim: Generating safety-critical scenarios for self-driving vehicles," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 11
[124] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, "Generating useful accident-prone driving scenarios via a learned traffic prior," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 11
[125] N. Hanselmann, K. Renz, K. Chitta, A. Bhattacharyya, and A. Geiger, "King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients," *arXiv.org*, vol. 2204.13683, 2022. 11
[126] A. Zhao, T. He, Y. Liang, H. Huang, G. V. den Broeck, and S. Soatto, "Sam: Squeeze-and-mimic networks for conditional visual driving policy learning," in *Proc. Conf. on Robot Learning (CoRL)*, 2020. 11
[127] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *arXiv.org*, vol. 2201.03545, 2022. 14
[128] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 14
[129] M. Li, Y. Wang, and D. Ramanan, "Towards streaming perception," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 15

Kashyap Chitta is a Ph.D. student with the Autonomous Vision Group led by Prof. Andreas Geiger, part of the Max Planck Institute for Intelligent Systems and University of Tübingen, Germany. He obtained his M.S. in Computer Vision in 2018 from Carnegie Mellon University under the supervision of Prof. Martial Hebert. During this time, he also worked as a Deep Learning Intern at NVIDIA on two occasions.

Aditya Prakash is a Ph.D. student at the University of Illinois Urbana Champaign, advised by Prof. Saurabh Gupta. Prior to this, he worked as a research assistant at the Max Planck Institute for Intelligent Systems Tübingen as part of the Autonomous Vision Group led by Prof. Andreas Geiger. He obtained his bachelors at the Indian Institute of Technology Roorkee and has worked as an intern at Adobe Research, Indian Institute of Science and NAVER Clova AI Research.



Bernhard Jaeger is a Ph.D. student with the Autonomous Vision Group led by Prof. Andreas Geiger, part of the International Max Planck Research School for Intelligent Systems and University of Tübingen, Germany. He obtained his M.Sc. in Computer Science in 2021 from University of Tübingen and his B.Sc. degree in Informatics: Games Engineering from the Technical University of Munich in 2018. He worked for 1 year at FERCHAU GmbH as a software developer.



Zehao Yu is a Ph.D. student with the Autonomous Vision Group led by Prof. Andreas Geiger, at the University of Tübingen, Germany. He obtained his M.S. in Computer Science in 2021 from ShanghaiTech University under the supervision of Prof. Shenghua Gao and his B.Sc. degree in Software Engineering from Xiamen University.

Katrin Renz is a Ph.D. student with the Autonomous Vision Group led by Prof. Andreas Geiger, part of the Max Planck Institute for Intelligent Systems and University of Tübingen, Germany. She obtained her masters in 2021 from the University of Heilbronn and has spent time as a visiting student in the Visual Geometry Group at the University of Oxford.

Andreas Geiger is a professor at the University of Tübingen. Prior to this, he was a visiting professor at ETH Zürich and a group leader at the Max Planck Institute for Intelligent Systems. He studied at KIT, EPFL and MIT, and received his PhD degree in 2013 from the Karlsruhe Institute of Technology (KIT). His research interests are at the intersection of computer vision, machine learning and robotics, with a particular focus on 3D scene perception, deep representation learning, generative models and sensori-motor control in the context of autonomous systems. He maintains the KITTI and KITTI-360 benchmarks.