PlanT: Explainable Planning Transformers via Object-Level Representations

Katrin Renz[1,2] **Kashyap Chitta**[1,2] **Otniel-Bogdan Mercea**[1]
A. Sophia Koepke[1] **Zeynep Akata**[1,2,3] **Andreas Geiger**[1,2]
[1]University of Tübingen [2]Max Planck Institute for Intelligent Systems, Tübingen
[3]Max Planck Institute for Informatics, Saarbrücken
https://www.katrinrenz.de/plant

Abstract: Planning an optimal route in a complex environment requires efficient reasoning about the surrounding scene. While human drivers prioritize important objects and ignore details not relevant to the decision, learning-based planners typically extract features from dense, high-dimensional grid representations containing all vehicle and road context information. In this paper, we propose PlanT, a novel approach for planning in the context of self-driving that uses a standard transformer architecture. PlanT is based on imitation learning with a compact object-level input representation. On the Longest6 benchmark for CARLA, PlanT outperforms all prior methods (matching the driving score of the expert) while being 5.3× faster than equivalent pixel-based planning baselines during inference. Combining PlanT with an off-the-shelf perception module provides a sensor-based driving system that is more than 10 points better in terms of driving score than the existing state of the art. Furthermore, we propose an evaluation protocol to quantify the ability of planners to identify relevant objects, providing insights regarding their decision-making. Our results indicate that PlanT can focus on the most relevant object in the scene, even when this object is geometrically distant.

Keywords: Autonomous Driving, Transformers, Explainability

1 Introduction

The ability to plan is an important aspect of human intelligence, allowing us to solve complex navigation tasks. For example, to change lanes on a busy highway, a driver must wait for sufficient space in the new lane and adjust the speed based on the expected behavior of the other vehicles. Humans quickly learn this and can generalize to new scenarios, a trait we would also like autonomous agents to have. Due to the difficulty of the planning task, the field of autonomous driving is shifting away from traditional rule-based algorithms [1, 2, 3, 4, 5, 6, 7, 8] towards learning-based solutions [9, 10, 11, 12, 13, 14]. Learning-based planners directly map the environmental state representation (e.g., HD maps and object bounding boxes) to waypoints or vehicle controls. They emerged as a scalable alternative to rule-based planners which require significant manual effort to design.

Interestingly, while humans reason about the world in terms of objects [15, 16, 17], most existing learned planners [9, 12, 18] choose a high-dimensional pixel-level input representation by rendering bird's eye view (BEV) images of detailed HD maps (Fig. 1 left). It is widely believed that this kind of accurate scene understanding is key for robust self-driving vehicles, leading to significant interest in recovering pixel-level BEV information from sensor inputs [19, 20, 21, 22, 23, 24]. In this paper, we investigate whether such detailed representations are actually necessary to achieve convincing planning performance. We propose PlanT, a learning-based planner that leverages an object-level representation (Fig. 1 right) as an input to a transformer encoder [25]. We represent a scene as a set of features corresponding to (1) nearby vehicles and (2) the route the planner must follow. We show that despite the low feature dimensionality, our model achieves state-of-the-art results. We then propose a novel evaluation scheme and metric to analyze explainability which is generally applicable to any learning-based planner. Specifically, we test the ability of a planner to identify the objects that are the most relevant to account for to plan a collision-free route.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

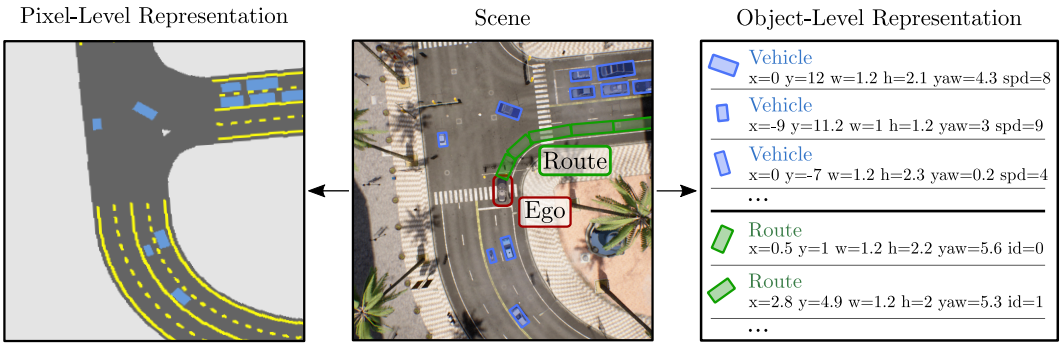


Figure 1: **Scene Representations for Planning.** As an alternative to the dominant paradigm of pixel-level planners (left), we show the effectiveness of compact object-level representations (right).

We perform a detailed empirical analysis of learning-based planning on the *Longest6* benchmark [26] of the CARLA simulator [27]. We first identify the key missing elements in the design of existing learned planners such as their incomplete field of view and sub-optimal dataset and model sizes. We then show the advantages of our proposed transformer architecture, including improvements in performance and significantly faster inference times. Finally, we show that the attention weights of the transformer, which are readily accessible, can be used to represent object relevance. Our qualitative and quantitative results on explainability confirm that PlanT attends to the objects that match our intuition for the relevance of objects for safe driving.

Contributions. (1) Using a simple object-level representation, we significantly improve upon the previous state of the art for planning on CARLA via PlanT, our novel transformer-based approach. (2) Through a comprehensive experimental study, we identify that the ego vehicle's route, a full 360° field of view, and information about vehicle speeds are critical elements of a planner's input representation. (3) We propose a protocol and metric for evaluating a planner's prioritization of obstacles in a scene and show that PlanT is more explainable than CNN-based methods, i.e., the attention weights of the transformer identify the most relevant objects more reliably.

2 Related Work

Intermediate Representations for Driving. Early work on decoupling end-to-end driving into two stages predicts a set of low-dimensional *affordances* from sensor inputs with CNNs which are then input to a rule-based planner [28]. These affordances are scene-descriptive attributes (e.g. emergency brake, red light, center-line distance, angle) that are compact, yet comprehensive enough to enable simple driving tasks, such as urban driving on the initial version of CARLA [27]. Unfortunately, methods based on affordances perform poorly on subsequent benchmarks in CARLA which involve higher task complexity [29]. Most state-of-the-art driving models instead rely heavily on annotated 2D data either as intermediate representations or auxiliary training objectives [26, 30]. Several subsequent studies show that using semantic segmentation as an intermediate representation helps for navigational tasks [31, 32, 33, 34]. More recently, there has been a rapid growth in interest in using BEV semantic segmentation maps as the input representation to planners [9, 12, 30, 18]. To reduce the immense labeling cost of such segmentation methods, Behl et al. [35] propose visual abstractions, which are label-efficient alternatives to dense 2D semantic segmentation maps. They show that reduced class counts and the use of bounding boxes instead of pixel-accurate masks for certain classes are sufficient. Wang et al. [36] explore the use of object-centric representations for planning by explicitly extracting objects and rendering them into a BEV input for a planner. However, so far, the literature lacks a systematic analysis of whether object-centric representations are better or worse than BEV context techniques for planning in dense traffic, which we address in this work. We keep our representation simple and compact by directly considering the set of objects as inputs to our models. In addition to baselines using CNNs to process the object-centric representation, we show that using a transformer leads to improved performance, efficiency, and explainability.

Transformers for Forecasting. Transformers obtain impressive results in several research areas [25, 37, 38, 39], including simple interactive environments such as Atari games [40, 41, 42, 43, 44]. While the end objective differs, one application domain that involves similar challenges

to planning is motion forecasting. Most existing motion forecasting methods use a rasterized input in combination with a CNN-based network architecture [45, 46, 47, 48, 49, 50]. Gao et al. [51] show the advantages of object-level representations for motion forecasting via Graph Neural Networks (GNN). Several follow-ups to this work use object-level representations in combination with Transformer-based architectures [52, 53, 54]. Our key distinctions when compared to these methods are the architectural simplicity of PlanT (our use of simple self-attention transformer blocks and the proposed route representation) as well as our closed-loop evaluation protocol (we evaluate the driving performance in simulation and report online driving metrics).

Explainability. Explaining the decisions of neural networks is a rapidly evolving research field [55, 56, 57, 58, 59, 60, 61]. In the context of self-driving cars, existing work uses text [62] or heatmaps [63] to explain decisions. In our work, we can directly obtain post hoc explanations for decisions of our learning-based PlanT architecture by considering its learned attention. While the concurrent work CAPO [64] uses a similar strategy, it only considers pedestrian-ego interactions on an empty route, while we consider the full planning task in an urban environment with dense traffic. Furthermore, we introduce a simple metric to measure the quality of explanations for a planner.

3 Planning Transformers

In this section, we provide details about our task setup, novel scene representation, simple but effective architecture, and training strategy resulting in state-of-the-art performance. A PyTorch-style pseudo-code snippet outlining PlanT and its training is provided in the supplementary material.

Task. We consider the task of point-to-point navigation in an urban setting where the goal is to drive from a start to a goal location while reacting to other dynamic agents and following traffic rules. We use Imitation Learning (IL) to train the driving agent. The goal of IL is to learn a policy π that imitates the behavior of an expert π^* (the expert implementation is described in Section 4). In our setup, the policy is a mapping $\pi : \mathcal{X} \rightarrow \mathcal{W}$ from our novel object-level input representation $\mathcal{X}$ to the future trajectory $\mathcal{W}$ of an expert driver. For following traffic rules, we assume access to the state of the next traffic light relevant to the ego vehicle $l \in \{\text{green}, \text{red}\}$.

Tokenization. To encode the task-specific information required from the scene, we represent it using a set of objects, with vehicles and segments of the route each being assigned an oriented bounding box in BEV space (Fig. 1 right). Let $\mathcal{X}_t = \mathcal{V}_t \cup \mathcal{S}_t$, where $\mathcal{V}_t \in \mathbb{R}^{V_t \times A}$ and $\mathcal{S}_t \in \mathbb{R}^{S_t \times A}$ represent the set of vehicles and the set of route segments at time-step t with $A = 6$ attributes each. Specifically, if $\mathbf{o}_{i,t} \in \mathcal{X}_t$ represents a particular object, the attributes of $\mathbf{o}_{i,t}$ include an object type-specific attribute $z_{i,t}$ (described below), the position of the bounding box $(x_{i,t}, y_{i,t})$ relative to the ego vehicle, the orientation $\varphi_{i,t} \in [0, 2\pi]$, and the extent $(w_{i,t}, h_{i,t})$. Thus, each object $\mathbf{o}_{i,t}$ can be described as a vector $\mathbf{o}_{i,t} = \{z_{i,t}, x_{i,t}, y_{i,t}, \varphi_{i,t}, w_{i,t}, h_{i,t}\}$, or concisely as $\{\mathbf{o}_{i,t,a}\}_{a=1}^{6}$.

For the vehicles $\mathcal{V}_t$, we extract the attributes directly from the simulator in our main experiments and use an off-the-shelf perception module based on CenterNet [65] (described in the supplementary material) for experiments involving a full driving system. We consider only vehicles up to a distance D_{max} from the ego vehicle, and use $\mathbf{o}_{i,t,1}$ (i.e., $z_{i,t}$) to represent the speed.

To obtain the route segments $\mathcal{S}_t$, we first sample a dense set of N_t points $\mathcal{U}_t \in \mathbb{R}^{N_t \times 2}$ along the route ahead of the ego vehicle at time-step t. We directly use the ground-truth points from CARLA as $\mathcal{U}_t$ in our main experiments and predict them with a perception module for the PlanT with perception experiments in Section 4.2. The points are subsampled using the Ramer-Douglas-Peucker algorithm [66, 67] to select a subset $\hat{\mathcal{U}}_t$. One segment spans the area between two points subsampled from the route, $\mathbf{u}_{i,t}, \mathbf{u}_{i+1,t} \in \hat{\mathcal{U}}_t$. Specifically, $\mathbf{o}_{i,t,1}$ (i.e., $z_{i,t}$) denotes the ordering for the current time-step t, starting from 0 for the segment closest to the ego vehicle. We set the segment length $\mathbf{o}_{i,t,6} = ||\mathbf{u}_{i,t} - \mathbf{u}_{i+1,t}||_2$, and the width, $\mathbf{o}_{i,t,5}$, equal to the lane width. In addition, we clip $\mathbf{o}_{i,t,6} \leq L_{max}, \forall i, t$; and always input a fixed number of segments N_s to our policy. More details and visualizations of the route representation are provided in the supplementary material.

Token Embeddings. Our model is illustrated in Fig. 2. As a first step, applying a transformer backbone requires the generation of embeddings for each input token, for which we define a linear projection $\rho : \mathbb{R}^6 \rightarrow \mathbb{R}^H$ (where H is the desired hidden dimensionality). To obtain token embeddings $\mathbf{e}_{i,t}$, we add the projected input tokens $\mathbf{o}_{i,t}$ to a learnable object type embedding vector $\mathbf{e}_v \in \mathbb{R}^H$ or $\mathbf{e}_s \in \mathbb{R}^H$, indicating to which type the token belongs (vehicle or route segment).

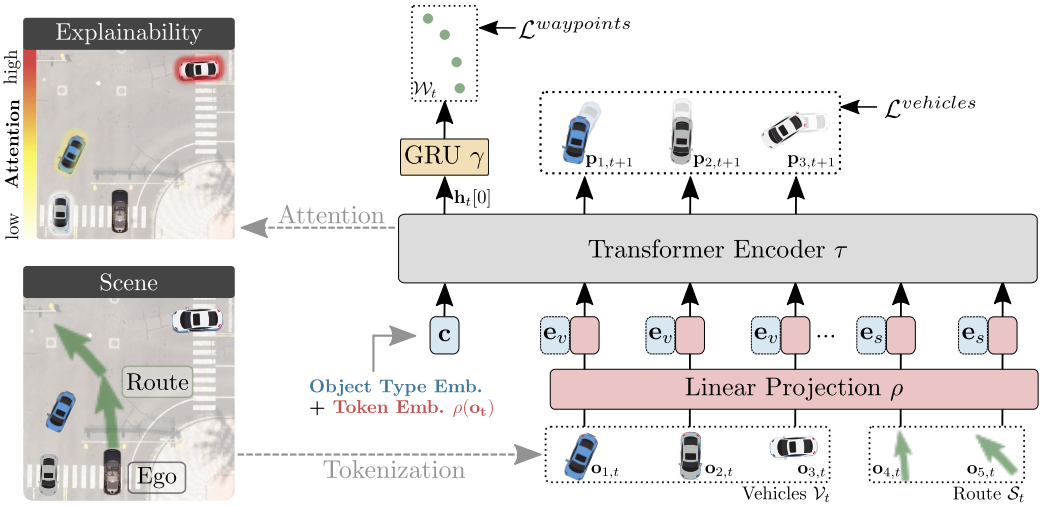


Figure 2: **Planning Transformer (PlanT).** We represent a scene (bottom left) using a set of objects containing the vehicles and route to follow (green arrows). We embed these via a linear projection (bottom right) and process them with a transformer encoder. PlanT outputs future waypoints with a GRU decoder. We use a self-supervised auxiliary task of predicting the future of other vehicles. Further, extracting and visualizing the attention weights yields an explainable decision (top left).

$$\mathbf{e}_{i,t} = \begin{cases} \rho(\mathbf{o}_{i,t}) + \mathbf{e}_v & \forall \quad \mathbf{o}_{i,t} \in \mathcal{V}_t, \\ \rho(\mathbf{o}_{i,t}) + \mathbf{e}_s & \forall \quad \mathbf{o}_{i,t} \in \mathcal{S}_t. \end{cases} \tag{1}$$

Main Task: Waypoint Prediction. The main building block for the IL policy π is a standard transformer encoder, $\tau : \mathbb{R}^{V_t+S_t+1\times H} \rightarrow \mathbb{R}^{V_t+S_t+1\times H}$, based on the BERT architecture [37]. Specifically, we define a learnable `[CLS]` token, $\mathbf{c} \in \mathbb{R}^H$ (based on [37, 39]) and stack this with other token embeddings to obtain the transformer input $[\mathbf{c}, \mathbf{e}_{1,t}, ..., \mathbf{e}_{V_t+S_t,t}]$. The `[CLS]` token's processing through τ involves an attention-based aggregation of the features from all other tokens, after which it is used for generating the waypoint predictions via an auto-regressive waypoint decoder, $\gamma : \mathbb{R}^{(H+1)} \rightarrow \mathbb{R}^{W\times 2}$. For a detailed description of the waypoint decoder architecture, see [18, 26]. We concatenate the binary traffic light flag, l_t to the transformer output as the initial hidden state to the decoder which makes use of GRUs [68] to predict the future trajectory $\mathcal{W}_t$ of the ego vehicle, centered at the coordinate frame of the current time-step t. The trajectory is represented by a sequence of 2D waypoints in BEV space, $\{\mathbf{w}_w = (x_w, y_w)\}_{w=t+1}^{t+W}$ for $W = 4$ future time-steps:

$$\mathcal{W}_t = \gamma(l_t, \mathbf{h}_t[0]), \quad \text{where } \mathbf{h}_t = \tau([\mathbf{c}, \mathbf{e}_{1,t}, ..., \mathbf{e}_{V_t+S_t,t}]). \tag{2}$$

Auxiliary Task: Vehicle Future Prediction. In addition to the primary waypoint prediction task, we propose the auxiliary task of predicting the future attributes of other vehicles. This is aligned with the overall driving goal in two ways. (1) The ability to reason about the future of other vehicles is important in an urban environment as it heavily influences the ego vehicle's own future. (2) Our main task is to predict the ego vehicle's future trajectory, which means the output feature of the transformer needs to encode all the information necessary to predict the future. Supervising the outputs of all vehicles on a similar task (i.e., predicting vehicle poses at a future time-step) exploits synergies between the task of the ego vehicle and the other vehicles [69, 30]. Specifically, using the output embeddings $\{\mathbf{h}_{i,t}\}_{i=1}^{V_t}$ corresponding to all vehicle tokens $\mathbf{o}_{i,t} \in \mathcal{V}_t$, we predict class probabilities $\{\{\mathbf{p}_{i,t+1,a}\}_{a=1}^{6}\}_{i=1}^{V_t}$ for the speed, position, orientation, and extent attributes from the next time-step $\{\mathbf{o}_{i,t+1,a}\}_{a=1}^{6}$ using a linear layer per attribute type $\{\psi_a : \mathbb{R}^H \rightarrow \mathbb{R}^{Z_a}\}_{a=1}^{6}$:

$$\mathbf{p}_{i,t+1,a} = \text{Softmax}\left(\psi_a(\mathbf{h}_{i,t})\right), \quad \text{where } a = 1, ..., 6. \tag{3}$$

We choose to discretize each attribute into Z_a bins to allow for uncertainty in the predictions since the future is multi-modal. This is also better aligned with how humans drive without predicting exact locations and velocities, where a rough estimate is sufficient to make a safe decision.

Loss Functions. Following recent driving models [9, 11, 30, 26], we leverage the L_1 loss to the ground truth future waypoints $\mathbf{w}^{gt}$ as our main training objective. For the auxiliary task, we calculate

the cross-entropy loss $\mathcal{L}_{CE}$ using a one-hot encoded representation $\mathbf{p}_{i,t+1}^{gt}$ of the ground truth future vehicle attributes $\mathbf{o}_{i,t+1}^{gt}$. We train the model in a multi-task setting using a weighted combination of these losses with a weighting factor λ:

$$\mathcal{L} = \underbrace{\frac{1}{W}\sum_{w=1}^{W} ||\mathbf{w}_w - \mathbf{w}_w^{gt}||_1}_{\mathcal{L}^{waypoints}} + \underbrace{\frac{\lambda}{V}\sum_{i=1}^{V_t}\sum_{a=1}^{6} \mathcal{L}_{CE}\left(\mathbf{p}_{i,t+1,a}, \mathbf{p}_{i,t+1,a}^{gt}\right)}_{\mathcal{L}^{vehicles}}. \tag{4}$$

4 Experiments

In this section, we describe our experimental setup, evaluate the driving performance of our approach, analyze the explainability of its driving decisions, and finally discuss limitations.

Dataset and Benchmark. We use the expert, dataset and evaluation benchmark *Longest 6* proposed by [26]. The expert policy is a rule-based algorithm with access to ground truth locations of the vehicles as well as privileged information that is not available to PlanT such as their actions and dynamics. Using this information, the expert determines the future position of all vehicles and estimates intersections between its own future position and those of the other vehicles to prevent most collisions. The dataset collected with this expert contains 228k frames. We use this as our reference point denoted by $1\times$. For our analysis, we also generate additional data following [26] but with different initializations of the traffic. The data quantities we use are always relative to the original dataset (i.e., $2\times$ contains double the data, $3\times$ contains triple). We refer the reader to [26] for a detailed description of the expert algorithm and dataset collection.

Metrics. We report the established metrics of the CARLA leaderboard [70]: *Route Completion (RC)*, *Infraction Score (IS)*, and *Driving Score (DS)*, which is the weighted average of the RC and IS. In addition, we show *Collisions with Vehicles per kilometer (CV)* and *Inference Time (IT)* for one forward pass of the model, measured in milliseconds on a single RTX 3080 GPU.

Baselines. To highlight the advantages of learning-based planning, we include a **rule-based** planning baseline that uses the same inputs as PlanT. It follows the same high-level algorithm as the expert but estimates the future of other vehicles using a constant speed assumption since it does not have access to their actions. **AIM-BEV** [18] is a recent privileged agent trained using IL. It uses a BEV semantic map input with channels for the road, lane markings, vehicles, and pedestrians, and a GRU identical to PlanT to predict a trajectory for the ego vehicle which is executed using lateral and longitudinal PID controllers. **Roach** [12] is a Reinforcement Learning (RL) based agent with a similar input representation as AIM-BEV that directly outputs driving actions. Roach and AIM-BEV are the closest existing methods to PlanT. However, they use a different input field of view in their representation leading to sub-optimal performance. We additionally build **PlanCNN**, a more competitive CNN-based approach for planning with the same training data and input information as PlanT, which is adapted from AIM-BEV to input a rasterized version of our object-level representation. We render the oriented vehicle bounding boxes in one channel, represent the speed of each pixel in a second channel, and render the oriented bounding boxes of the route in the third channel. We provide detailed descriptions of the baselines in the supplementary material.

Implementation. Our analysis includes three BERT encoder variants taken from [71]: `MINI`, `SMALL`, and `MEDIUM` with 11.2M, 28.8M and 41.4M parameters respectively. For PlanCNN, we experiment with two backbones: ResNet-18 and ResNet-34. We choose these architectures to maintain an IT which enables real-time execution. We train the models from scratch on 4 RTX 2080Ti GPUs with a total batch size of 128. Optimization is done with AdamW [72] for 47 epochs with an initial learning rate of 10^{-4} which we decay by 0.1 after 45 epochs. Training takes approximately 2.8, 3.4, and 4 hours for the three BERT variants on the $1\times$ dataset. We set the weight decay to 0.1 and clip the gradient norm at 1.0. For the auxiliary objective, we use quantization precisions Z_a of 128 bins for the position, 4 bins for the speed and 32 bins for the orientation of the vehicles. We use $T_{in} = 0$ and $\delta t = 1$ for auxiliary supervision. The loss weight λ is set to 0.2. By default, we use $D_{max} = 30\,\mathrm{m}$, $N_s = 2$, and $L_{max} = 10\,\mathrm{m}$. For our experiment with a full driving stack, we use a perception module based on TransFuser [26] to obtain the object-level input representation for PlanT. Additional details regarding this perception module as well as detailed ablation studies on the multi-task training and input representation hyperparameters are provided in the supplementary material.

	Method	Input	DS ↑	RC ↑	IS ↑	CV ↓	IT ↓
Sensor	LAV* [30]	Camera + LiDAR	32.74±1.45	70.36±3.14	0.51±0.02	**0.84±0.11**	-
	TransFuser [26]	Camera + LiDAR	47.30±5.72	**93.38±1.20**	0.50±0.60	2.44±0.64	101.24
	PlanT w/ perception	Camera + LiDAR	**57.66±5.01**	88.20±0.94	**0.65±0.06**	0.97±0.09	37.61
Privileged	Rule-based	Obj. + Route	38.00±1.64	29.09±2.12	0.84±0.00	0.64±0.07	-
	AIM-BEV [18]	Rast. Obj. + HD Map	45.06±1.68	78.31±1.12	0.55±0.01	1.67±0.16	18.14
	Roach* [12]	Rast. Obj. + HD Map	55.27±1.43	88.16±1.52	0.62±0.02	0.76±0.07	3.24
	PlanCNN	Rast. Obj. + Rast. Route	77.47±1.34	**94.53±2.59**	0.81±0.03	0.43±0.05	28.94
	PlanT	Obj. + Route	**81.36±6.54**	93.55±2.62	**0.87±0.05**	**0.31±0.12**	10.79
	Expert [26]	*Obj. + Route + Actions*	*76.91±2.23*	*88.67±0.56*	*0.86±0.03*	*0.28±0.06*	-

Table 1: **Longest6 Results.** We show the mean±std for 3 evaluations. PlanT reaches expert-level performance and requires significantly less inference time than the baselines. *We evaluate the publicly available checkpoints at the time of submission for LAV and Roach. See supplementary material for results with checkpoints released later.

4.1 Obtaining Expert-Level Driving Performance

In the following, we discuss the key findings of our study which enable expert-level driving with learned planners. We begin with a discussion of the privileged methods and analyze the sensor-based methods in Section 4.2. Unless otherwise specified, the experiments consider the largest version of our dataset (3×) and models (`MEDIUM` for PlanT, ResNet-34 for PlanCNN).

Input Representation. Table 1 compares the performance on the Longest6 benchmark. The rule-based system acts cautiously and gets blocked often. Among the learning-based methods, both PlanCNN and PlanT significantly outperform AIM-BEV [18] and Roach [12]. We systematically break down the factors leading to this in Table 2a by studying the following: (1) the representation used for the road layout, (2) the horizontal field of view, (3) whether objects behind the ego vehicle are part of the representation, and (4) whether the input representation incorporates speed.

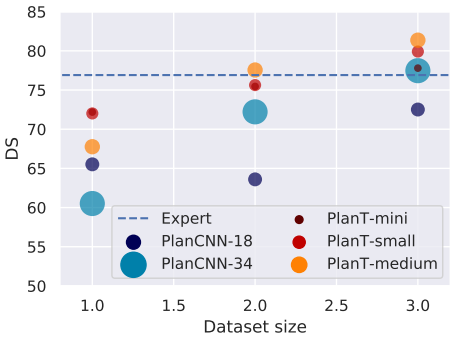
Roach uses the same view to the sides as AIM-BEV but additionally includes $8\,\mathrm{m}$ to the back and multiple input frames to reason about speed. We see in Table 2a that training PlanCNN in a configuration close to Roach (with the key differences being the removal of details from the map and a $0\,\mathrm{m}$ back view) results in a higher DS (59.97 vs. 55.27), demonstrating the importance of the route representation for urban driving. While additional information might be important when moving to more complex environments, our results suggest that the route is particularly important. Increasing the side view from 19.2 to $30\,\mathrm{m}$ improves PlanCNN from 59.97 to 70.72. Including vehicles to the rear further boosts PlanCNN's DS to 77.47 and improves PlanT's DS from 72.86 to 81.36. These results show that a full $360°$ field of view is helpful to handle certain situations encountered during our evaluation (e.g. changing lanes). Finally, removing the vehicle speed input significantly reduces the DS for both PlanCNN and PlanT (Table 2a), showing the importance of cues regarding motion.

Scaling. In Table 2b, we show the impact of scaling the dataset size and the model size for PlanT and PlanCNN. The circle size indicates the inference time (IT). First, we observe that PlanT demonstrates better data efficiency than PlanCNN, e.g., using the $1\times$ data setting is sufficient to reach the same performance as PlanCNN with $2\times$. Interestingly, scaling the data from $1\times$ to $3\times$ leads to expert-level performance, showing the effectiveness of scaling. In fact, $\text{PlanT}_{\texttt{MEDIUM}}$ (81.36) outperforms the expert (76.91) in some evaluation runs. We visualize one consistent failure mode of the expert that leads to this discrepancy in Fig. 3a. We observe that the expert sometimes stops once it has already entered an intersection if it anticipates a collision, which then leads to collisions or blocked traffic. On the other hand, PlanT learns to wait further outside an intersection before entering which is a smoother function than the discrete rule-based expert and subsequently avoids these infractions. Importantly, in our final setting, $\text{PlanT}_{\texttt{MEDIUM}}$ is around $3\times$ as fast as PlanCNN while being 4 points better in terms of the DS and $\text{PlanT}_{\texttt{MINI}}$ is $5.3\times$ as fast (IT=$5.46\,\mathrm{ms}$) while reaching the same DS as PlanCNN. This shows that PlanT is suitable for systems where fast inference time is a requirement. We report results with multiple training seeds in the supplementary material.

Loss. A detailed study of the training strategy for PlanT can be found in the supplementary material, where we show that the auxiliary loss proposed in Eq. (4) is crucial to its performance. However, since this is a self-supervised objective, it can be incorporated without additional annotation costs. This is in line with recent findings on training transformers that show the effectiveness of supervising multiple output tokens instead of just a single `[CLS]` token [73].

Method	Map	Side (m)	Back (m)	Speed	DS ↑
AIM-BEV [18]	✓	19.2	0	-	45.06±1.68
Roach [12]	✓	19.2	8	✓	55.27±1.43
PlanCNN	-	19.2	0	✓	59.97±4.47
	-	30	0	✓	70.72±2.99
	-	30	30	✓	77.47±1.34
	-	30	30	-	69.13±1.43
PlanT	-	30	0	✓	72.86±5.56
	-	30	30	✓	**81.36±6.54**
	-	30	30	-	72.34±3.30

(a) **Input Representation.** DS on Longest6 (3 evaluation runs) with different input properties.



(b) **Scaling.** Mean DS on Longest6 for different dataset and model sizes. Circle size shows IT.

Table 2: We investigate the choices of the input representation (Table 2a) and architecture (Table 2b). for learning-based planners. Including vehicles to the back of the ego, encoding vehicle speeds, and scaling to large models/datasets is crucial for the performance of both PlanCNN and PlanT.

4.2 Combining an Off-the-Shelf Perception Module with PlanT

Next, we discuss the results of the sensor-based methods in Table 1. We compare the proposed approach to LAV [30] and TransFuser [26], which are recent state-of-the-art sensor-based methods. Our perception module is based on TransFuser, enabling a fair comparison to this approach. Therefore, PlanT with perception only detects vehicles to its front and has a limited view to the sides (16m instead of 30m). Our approach outperforms TransFuser [26] by 10.36 points and LAV by 24.92 points. While TransFuser uses an ensemble and manually designed heuristics to creep forward if stuck [26], these are unnecessary for PlanT with perception. Since we do not use an ensemble, we observe a 2.7× speedup (101.24 ms vs. 37.61 ms) in IT compared to TransFuser. We refer to the supplementary material for a more detailed analysis.

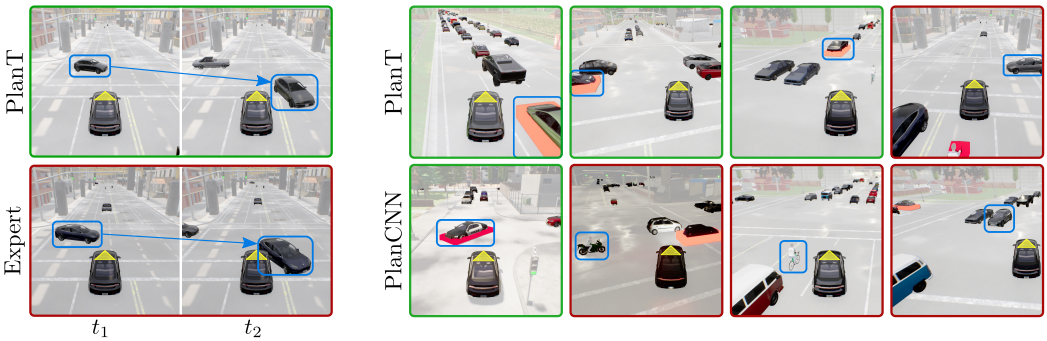
4.3 Explainability: Identification of Most Relevant Objects

Finally, we investigate the explainability of PlanT and PlanCNN by analyzing the objects in the scene that are relevant and crucial for the agent's decision. In particular, we measure the relevance of an object in terms of the learned attention for PlanT and by considering the impact that the removal of each object has on the output predictions for PlanCNN. To quantify the ability to reason about the most relevant objects, we propose a novel evaluation scheme together with the *Relative Filtered Driving Score (RFDS)*. For the rule-based expert algorithm, collision avoidance depends on a single vehicle which it identifies as the reason for braking. To measure the RFDS of a learned planner, we run one forward pass of the planner (without executing the actions) to obtain a scalar *relevance score* for each vehicle in the scene. We then execute the expert algorithm while restricting its observations to the (single) vehicle with the highest relevance score. The RFDS is defined as the relative DS of this restricted version of the expert compared to the default version which checks for collisions against all vehicles. We describe the extraction of the relevance score for PlanT and PlanCNN in the following. Our protocol leads to a fair comparison of different agents as the RFDS does not depend on the ability to drive itself but only on the obtained ranking of object relevance.

Baselines. As a naïve baseline, we consider the inverse distance to the ego vehicle as a vehicle's relevance score, such that the expert only sees the closest vehicle. For PlanT, we extract the relevance score by adding the attention weights of all layers and heads for the `[CLS]` token. This only requires a single forward pass of PlanT. Since PlanCNN does not use attention, we choose a masking method to find the most salient region in the image, using the same principle as [59, 58, 57]. We remove one object at a time from the input image and compute the L_1 distance to the predicted waypoints for the full image. The objects are then ranked based on how much their absence affects the L_1 distance.

Method	RFDS ↑
Inverse Distance	29.13±0.54
PlanCNN + Masking	82.83±6.79
PlanT + Attention	**96.82±2.12**

Table 3: **RFDS.** Relative score of the expert when only observing the most relevant vehicle according to the respective planner.



(a) **PlanT vs. Expert.** PlanT waits further outside the intersection than the expert to avoid a collision.

(b) **RFDS.** Vehicles with the highest relevance score are marked with a red bounding rectangle. We show examples for successful matching of relevance score and intuition (green frames) and failures (red frames).

Figure 3: We contrast a failure case of the expert to PlanT (Fig. 3a) and show the quality of the relevance scores (Fig. 3b). The ego vehicle is marked with a yellow triangle and vehicles that either lead to collisions or are intuitively the most relevant in the scene are marked with a blue box.

Results. We provide results for the reasoning about relevant objects in Table 3. Both planners significantly outperform the distance-based baseline, with PlanT obtaining a mean RFDS of 96.82 compared to 82.83 for PlanCNN. We show qualitative examples in Figure 3b where we highlight the vehicle with the highest relevance score using a red bounding rectangle. Both planners correctly identify the most important object in simple scenarios. However, PlanT is also able to correctly identify the most important object in complex scenes. When merging into a lane (examples 1 & 2 from the left) it correctly looks at the moving vehicles coming from the rear to avoid collisions. Example 3 shows advanced reasoning about dynamics. The two vehicles closer to the ego vehicle are moving away at a high speed and are therefore not as relevant. PlanT already pays attention to the more distant vehicle behind them as this is the one that it would collide with if it does not brake. One of the failures of PlanT we observe is that it sometimes allocates the highest attention to a very close vehicle behind itself (example 4) and misses the relevant object. PlanCNN has more prominent errors when there are a large number of vehicles in the scene or when merging lanes (examples 2 & 3). To better assess the driving performance and relevance scores we provide additional results in the supplementary video.

5 Conclusion

In this work, we take a step towards efficient, high-performance, explainable planning for autonomous driving with a novel object-level representation and transformer-based architecture called PlanT. Our experiments highlight the importance of correctly encoding the ego vehicle's route for planning. We show that incorporating a 360° field of view, information about vehicle speeds, and scaling up both the architecture and dataset size of a learned planner are essential to achieve state-of-the-art results. Additionally, PlanT significantly outperforms state-of-the-art end-to-end sensor-based models even with a noisy and incomplete input representation obtained via a perception module. Finally, we demonstrate that PlanT can reliably identify the most relevant object in the scene via a new metric and evaluation protocol that measure explainability.

Limitations. Firstly, the expert driver used in our IL-based training strategy does not achieve a perfect score (Table 1) and has certain consistent failure modes (Fig. 3a, more examples in the supplementary material). Human data collection to address this would be time-consuming (the 3× dataset used in our experiments contains around 95 hours of driving). Second, all our experiments are conducted in simulation. Real-world scenarios are more diverse and challenging. However, CARLA is a high-fidelity simulator actively used by many researchers for autonomous driving, and previous findings demonstrate that systems developed in simulators like CARLA can be transferred to the real world [31, 74, 75]. Finally, our experiment with perception (Section 4.2) uses a single off-the-shelf perception module that was not specifically optimized for PlanT, leading to sub-optimal performance. It is a well-known limitation of modular systems that downstream modules cannot recover easily from errors made by earlier modules. A thorough analysis of perception robustness and uncertainty encoding are important research directions beyond the scope of this work.

Acknowledgments

This work was supported by the BMWi (KI Delta Learning, project number: 19A19013O), the BMBF (Tübingen AI Center, FKZ: 01IS18039A), the DFG (SFB 1233, TP 17, project number: 276693517), by the ERC (853489 - DEXIM), and by EXC (number 2064/1 – project number 390727645). We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting K. Renz, K. Chitta and O.-B. Mercea. The authors also thank Niklas Hanselmann and Markus Flicke for proofreading and Bernhard Jaeger for helpful discussions.

References

[1] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer. Vision and navigation for the carnegie-mellon navlab. *PAMI*, 10(3):362–372, May 1988.

[2] E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen. The seeing passenger car 'vamors-p'. In *IV*, 1994.

[3] J. J. Leonard, J. P. How, S. J. Teller, M. Berger, S. Campbell, G. A. Fiore, L. Fletcher, E. Frazzoli, A. S. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. R. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. E. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. *JFR*, 25(10):727–774, 2008.

[4] W. Xu, J. Pan, J. Wei, and J. M. Dolan. Motion planning under uncertainty for on-road autonomous driving. In *ICRA*, 2014.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 2016.

[6] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong. Baidu apollo EM motion planner. *arXiv.org*, 1807.08048, 2018.

[7] 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 *ECCV*, 2020.

[8] A. Cui, A. Sadat, S. Casas, R. Liao, and R. Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. In *ICCV*, 2021.

[9] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *CoRL*, 2019.

[10] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019.

[11] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *CVPR*, 2021.

[12] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *ICCV*, 2021.

[13] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *CoRL*, 2021.

[14] 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*, 2109.13602, 2021.

[15] D. Marr. Vision: A computational investigation into the human representation and processing of visual information. *W. H. Freeman, San Francisco*, 1982.

[16] E. S. Spelke and K. D. Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007.

[17] S. P. Johnson. Object perception. In *Oxford Research Encyclopedia of Psychology*. Oxford University Press, 2018.

[18] 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*, 2204.13683, 2022.

[19] K. Chitta, A. Prakash, and A. Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *ICCV*, 2021.

[20] A. Saha, O. Mendez Maldonado, C. Russell, and R. Bowden. Translating Images into Maps. In *ICRA*, 2022.

[21] B. Zhou and P. Krähenbühl. Cross-view Transformers for real-time Map-view Semantic Segmentation. In *CVPR*, 2022.

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

[23] 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*, 2204.05088, 2022.

[24] 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*, 2203.17270, 2022.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[26] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, , and A. Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *PAMI*, 2022.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017.

[28] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, pages 2722–2730, 2015.

[29] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *ICCV*, 2019.

[30] D. Chen and P. Krähenbühl. Learning from all vehicles. In *CVPR*, 2022.

[31] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. In *CoRL*, 2018.

[32] A. Sax, B. Emi, A. R. Zamir, L. J. Guibas, S. Savarese, and J. Malik. Learning to navigate using mid-level visual priors. In *CoRL*, 2019.

[33] A. Mousavian, A. Toshev, M. Fiser, J. Kosecká, A. Wahid, and J. Davidson. Visual representations for semantic target driven navigation. In *ICRA*, 2019.

[34] B. Zhou, P. Krähenbühl, and V. Koltun. Does computer vision matter for action? *Science Robotics*, 4(30), 2019.

[35] A. Behl, K. Chitta, A. Prakash, E. Ohn-Bar, and A. Geiger. Label efficient visual abstractions for autonomous driving. In *IROS*, 2020.

[36] D. Wang, C. Devin, Q. Cai, P. Krähenbühl, and T. Darrell. Monocular plan view networks for autonomous driving. In *IROS*, 2019.

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv.org*, 2018.

[38] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[39] 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*, 2010.11929, 2020.

[40] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, 2021.

[41] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

[42] H. Furuta, Y. Matsuo, and S. S. Gu. Generalized decision transformer for offline hindsight information matching. *arXiv.org*, 2111.10364, 2021.

[43] Q. Zheng, A. Zhang, and A. Grover. Online decision transformer. *arXiv.org*, 2202.05607, 2022.

[44] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv.org*, 2205.06175, 2022.

[45] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *WACV*, 2020.

[46] S. Casas, W. Luo, and R. Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018.

[47] J. Hong, B. Sapp, and J. Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *CVPR*, 2019.

[48] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv.org*, 1910.05449, 2019.

[49] Y. Biktairov, M. Stebelev, I. Rudenko, O. Shliazhko, and B. Yangel. Prank: motion prediction based on ranking. In *NeurIPS*, 2020.

[50] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *CVPR*, 2020.

[51] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *CVPR*, June 2020.

[52] 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 *IROS*, 2020.

[53] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. J. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *ICLR*, 2022.

[54] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou. Multimodal motion prediction with stacked transformers. In *CVPR*, June 2021.

[55] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017.

[56] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *NeurIPS*, 2017.

[57] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.

[58] V. Petsiuk, A. Das, and K. Saenko. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*, 2018.

[59] R. Fong, M. Patrick, and A. Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, 2019.

[60] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[61] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[62] J. Kim, A. Rohrbach, T. Darrell, J. Canny, and Z. Akata. Textual explanations for self-driving vehicles. In *ECCV*, 2018.

[63] J. Kim and J. Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *ICCV*, 2017.

[64] R. McAllister, B. Wulfe, J. Mercat, L. Ellis, S. Levine, and A. Gaidon. Control-aware prediction objectives for autonomous driving. In *ICRA*, 2022.

[65] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv.org*, 1904.07850, 2019.

[66] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Comput. Graph. Image Process.*, 1:244–256, 1972.

[67] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122, 1973.

[68] 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.

[69] J. Zhang and E. Ohn-Bar. Learning by watching. In *CVPR*, June 2021.

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

[71] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv.org*, 1908.08962, 2019.

[72] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv.org*, 1711.05101, 2017.

[73] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick. Masked autoencoders are scalable vision learners. *arXiv.org*, 2111.06377, 2021.

[74] D. Wang, C. Devin, Q. Cai, F. Yu, and T. Darrell. Deep object centric policies for autonomous driving. In *ICRA*, 2019.

[75] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *ICRA*, 2021.