Learning an Explainable Planner for Autonomous Driving

Anonymous Author(s)
Affiliation
Address
email

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 of the scene 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. With this representation, we demonstrate that information regarding the ego vehicle's route provides sufficient context regarding the road layout for planning. On the challenging 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. 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 reliably 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]. 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 [14, 15, 16], most existing learned planners [9, 11, 17] 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 [18, 19, 20, 21, 22, 23]. 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 [24]. We represent a scene as a set of features corresponding to (1) nearby vehicles and (2) the route the planner must follow. Each feature is low-dimensional, e.g., position, extent, and orientation (details in Section 3). We show that despite the low feature dimensionality, our model achieves state-of-the-art results using this representation. 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.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

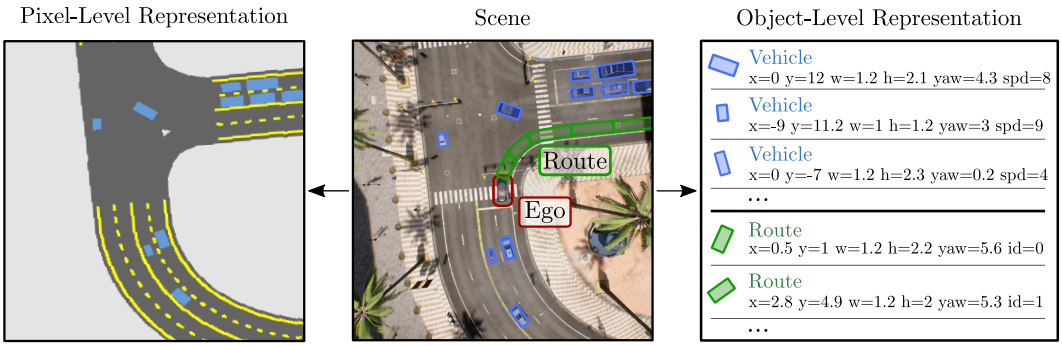


Figure 1: **Scene Representations for Planning.** As an alternative to the dominant paradigm of using detailed pixel-level representations for planning (left), we show the effectiveness of planners based on compact object-level representations (right).

We perform a detailed empirical analysis of learning-based planning on the *Longest6* benchmark [25] of the CARLA simulator [26]. 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. Addressing these issues leads to expert-level driving scores for planning with convolutional neural networks (CNNs), which is the dominant paradigm of this field, significantly advancing the state of the art. We then show the added advantages of our proposed transformer architecture, including further 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) We demonstrate that a simple object-level representation is sufficient to encode all the information relevant for planning in urban driving environments. (2) With this representation, we significantly improve upon the previous state of the art on CARLA when we use a standard CNN architecture, and obtain further gains in performance and runtime via PlanT, our novel transformer-based approach. (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 [27]. 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 [26]. Unfortunately, methods based on affordances perform poorly on subsequent benchmarks in CARLA which involve higher task complexity [28]. Most state-of-the-art driving models instead rely heavily on annotated 2D data either as intermediate representations or auxiliary training objectives [25, 29]. Several studies show that using semantic segmentation as an intermediate representation helps for navigational tasks [30, 31, 32, 33]. More recently, there has been a rapid growth in interest on using BEV semantic segmentation maps as the input representation to planners [9, 11, 29, 17]. To reduce the immense labeling cost of such segmentation methods, Behl et al. [34] 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 is sufficient. Wang et al. [35] explicitly extract objects and render them into a BEV image for planning. Instead of rendering objects onto a 2D image, we keep our representation low-dimensional and compact by directly considering the set of objects as inputs to our model. In addition, instead of using a CNN-based architecture to process the rendered images, we show that using a transformer leads to improved performance, efficiency, and explainability.

Transformers for Control. Transformers obtain impressive results in several research areas [24, 36, 37, 38]. To make use of this architecture, the input needs to be represented as a sequence or set. Using transformers by describing the environment state as a sequence has been successful for

Atari games [39, 40, 41, 42, 43]. In the field of motion forecasting for driving, Gao et al. [44] show the advantages of representing an HD map as a set of vectors instead of a BEV image. Several follow-up works use this representation in combination with transformer-based architectures [45, 46]. However, they only tackle motion forecasting and do not address the downstream planning task considered by our work. Moreover, while all of these methods rely on an HD map with detailed information regarding lane markings and road boundaries as inputs, we show that the ego vehicle's route is a more compact yet sufficient representation.

Explainability. Explaining the decisions of neural networks is a rapidly evolving research field [47, 48, 49, 50, 51, 52, 53]. Prior work on explaining policies focuses on reward decomposition [54], identifying the salient areas in an image using perturbations [55], approximating a deep neural network with a more interpretable model [56], or using interpretable attention mechanisms for predicting the correct action [57, 58]. In the context of self-driving cars, existing work uses text [59] or heatmaps [60] to explain decisions. In contrast, we can directly obtain post hoc explanations for decisions of our learning-based PlanT architecture by considering its learned attention. Furthermore, we introduce a simple approach to measure the quality of explanations for a planner by considering the change in driving performance when only considering the object that the explanation deemed most relevant in a scene for the planner's prediction.

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.

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^*. 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. We describe the expert and data collection procedure in Section 4. For following traffic rules, we assume access to the traffic light state $l \in \{\text{green}, \text{red}\}$ of the next traffic light relevant to the ego vehicle.

Object-level Representation. In order to drive safely, the input representation needs to encode all the task-specific information about the scene. To this end, we represent the scene as 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 $\mathcal{X} \in \mathbb{R}^{(V+R)\times A}$ be a set of V vehicles and R route segments with A attributes each. These attributes include the position of the bounding box (x_i, y_i) relative to the ego vehicle, the extent (w_i, h_i), the orientation $\varphi_i \in [0, 2\pi]$ and a type-specific attribute v_i, giving $A = 6$ dimensions. Each object can be described as a vector $\mathbf{o}_i = [x_i, y_i, w_i, h_i, \varphi_i, v_i]$. For a vehicle, we use v_i to represent the speed. For segments of the route, v_i denotes the ordering, starting from 0 for the segment closest to the ego vehicle. To obtain these route segments we subsample the dense set of points along the route provided by CARLA using the Ramer-Douglas-Peucker algorithm [61, 62]. One segment spans the area between two of these subsampled route points, with the length equal to the distance between the points and the width equal to the lane width. In addition, we restrict the maximum length of any single segment to L_{max} and always input a fixed number of segments N_s to our policy. Since the primary focus of our work is not perception but planning, we extract the object attributes directly from the simulator. We consider objects up to a distance D_{max} from the ego vehicle without any further filtering. Note that our representation contains similar information as that in the widely used BEV map representation (Fig. 1 left), but only encodes the route relevant to the ego vehicle instead of the whole map. In practice, we find that this is sufficient for current CARLA benchmarks (Table 2a). More details regarding the object extraction and visualizations of the route representation are provided in the supplementary material.

Output. We predict the future trajectory $\mathcal{W}$ 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, $\{\mathbf{w}_t = (x_t, y_t)\}_{t=1}^{T}$ for $T = 4$ future time-steps. These waypoints are provided to lateral and longitudinal PID controllers to output actions. Details are provided in the supplementary material.

Planning Transformer (PlanT). Our model is illustrated in Fig. 2. The main building block for the IL policy π is a standard transformer encoder, based on the BERT architecture [36]. While our architecture does not deviate from the original transformer implementation [24], adapting it to

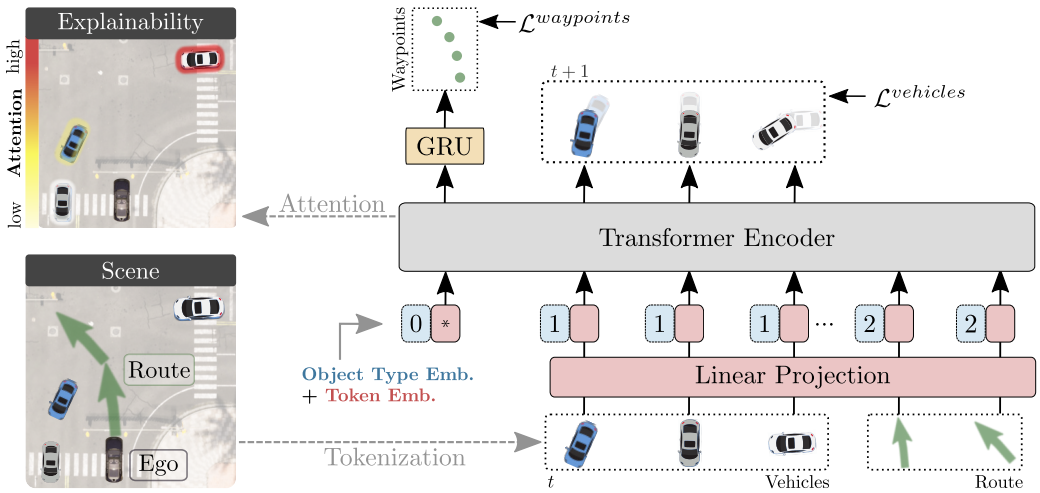


Figure 2: **Planning Transformer (PlanT).** We represent a scene (bottom left) using a set of objects containing the vehicles and route to follow (green arrows). We embed these via a linear projection (bottom right) and process them with a transformer encoder. PlanT outputs future waypoints with a GRU 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).

self-driving requires the tokenization of our proposed input representation. For this, we linearly project the attributes of each object to one token embedding. We then add the token embeddings to learnable object type embeddings indicating to which type the token belongs: `[CLS]`, vehicle or route. The `[CLS]` token (based on [36, 38]) is a learnable token prepended to the input. This token's processing involves an attention-based aggregation of the features from all other tokens, and is used for generating the waypoint predictions. At the final layer of the transformer, we extract the feature vector from the `[CLS]` token and reduce its dimensionality via a linear layer. As in previous work [17], we concatenate the binary traffic light flag and the feature vector before passing it to an auto-regressive waypoint decoder that makes use of GRUs [63]. We input the current ego position and goal location to the GRU, which predicts $T = 4$ differential waypoints $\{\delta \mathbf{w}_t\}_{t=1}^{T}$, where t is the future time-step. With these differential waypoints we obtain the future waypoints as $\{\mathbf{w}_t = \mathbf{w}_{t-1} + \delta \mathbf{w}_t\}_{t=1}^{T}$. For a detailed description of the waypoint decoder, see [17, 25].

Training. Following recent driving models [9, 29, 25], we leverage the L_1 loss to ground truth future waypoints $\mathbf{w}^{gt}$ as our main training objective. Besides this, 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 the location, pose and velocity of a future time-step) exploits synergies between the task of the ego vehicle and the other vehicles [64, 29]. We input the vehicle tokens for a given time-step T_{in} and predict class probabilities $\hat{\mathbf{p}}$ for every attribute of each vehicle of a future time-step $T_{in} + \delta t$ using a linear layer per attribute type on top of the corresponding output embedding. We choose to discretize the output attributes to allow uncertainty in the predictions since the future is non-deterministic. 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. We calculate the cross-entropy loss $\mathcal{L}_{CE}$ using the one-hot encoded representation of the attribute $\mathbf{p}^{gt}$ as the ground truth class label. We train the model in a multi-task setting using a weighted combination of these loss functions with a weighting factor λ:

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

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 [25]. 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 [25], but use different seeds to obtain 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 [25] for a detailed description of the expert algorithm and dataset collection.

Metrics. We report the established metrics of the CARLA leaderboard [65]: *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. A detailed description of the metrics is included in the supplementary material.

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** [17] 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** [11] 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 [66]: `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 [67] 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 of $0.5\,\mathrm{m}$ for the position and extent, $3.75\,\mathrm{km/h}$ for the speed and $11.25°$ 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 m, $N_s = 2$, and L_{max} =30 m. Additional details regarding the model configurations as well as detailed ablation studies on the multi-task training and input representation hyperparameters are provided in the supplementary material.

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. Unless otherwise specified, the experiments consider the largest version of our dataset ($3\times$) and models (`MEDIUM` for PlanT, ResNet-34 for PlanCNN).

We first analyze LAV [29] and TransFuser [25], which are recent state-of-the-art sensor-based methods on CARLA. Since PlanT accesses additional input information, it is not directly comparable to these baselines. However, we include them to put our results into perspective and to demonstrate the

Method	Input	DS ↑	RC ↑	IS ↑	CV ↓	IT ↓
Rule-based	Obj. + Route	29.09±2.12	38.00±1.64	0.84±0.00	0.64±0.07	-
LAV* [29]	Camera + LiDAR	32.74±1.45	70.36±3.14	0.51±0.02	0.84±0.11	-
TransFuser [25]	Camera + LiDAR	47.30±5.72	93.38±1.20	0.50±0.60	2.44±0.64	-
AIM-BEV* [17]	Rast. Obj. + HD Map	45.06±1.68	78.31±1.12	0.55±0.01	1.67±0.16	18.14±1.08
Roach* [11]	Rast. Obj. + HD Map	55.27±1.43	88.16±1.52	0.62±0.02	0.76±0.07	3.24±0.15
PlanCNN	Rast. Obj. + Rast. Route	77.47±1.34	**94.53±2.59**	0.81±0.03	0.43±0.05	28.94±1.24
PlanT	Obj. + Route	**81.36±6.54**	93.55±2.62	**0.87±0.05**	**0.31±0.12**	10.79±0.47
Expert [25]	*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 author-provided pre-trained models for LAV, AIM-BEV, and Roach.

potential gain for future sensor-based models. The results are shown in Table 1. Interestingly, both sensor-based methods outperform the rule-based planning algorithm despite their lack of ground-truth object and route information, showing the limitations of rule-based systems in complex environments. In contrast, PlanT is over 30 points better than TransFuser in terms of DS (81.36 vs. 47.30), combining the high IS of the rule-based approach and high RC of TransFuser. Importantly, this improvement is achieved using the same expert policy and IL-based training objective as TransFuser. Additionally, TransFuser incorporates manually designed heuristics in its controller to creep forward if the policy gets stuck without cause [25], which are unnecessary for PlanT.

Input Representation. In Table 1, we observe that both PlanCNN and PlanT significantly outperform AIM-BEV [17] and Roach [11]. 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.

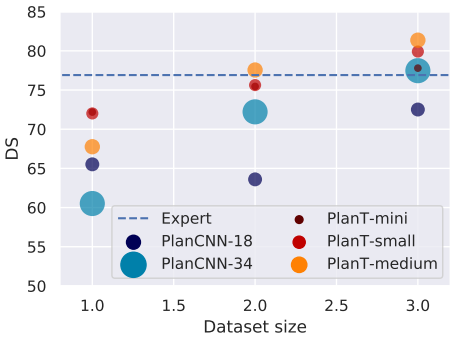
Roach uses the same maximum distance to the sides of the vehicle as AIM-BEV but additionally includes $8\,\mathrm{m}$ to the back and multiple input frames to reason about speed. We see in Table 2a that training PlanCNN in a configuration close to Roach (with the key differences being the removal of details from the map and a $0\,\mathrm{m}$ back view) results in a higher DS (59.97 vs. 55.27), showing that it is possible to replace the lane marking and road boundary information in the HD map with our compact representation. Increasing the side view from 19.2 to $30\,\mathrm{m}$ improves PlanCNN from 59.97 to 70.72 (Table 2a). This indicates that reasoning about distant vehicles is important. Furthermore, including vehicles to the rear further boosts PlanCNN's DS to 77.47. This also holds true for PlanT, where adding the rear improves the 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.

PlanT vs. PlanCNN. 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) needed for that model. 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 with all the models other than PlanCNN with ResNet-18, 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. To demonstrate the reproducibility of our results with different initializations, we report driving scores of PlanCNN and PlanT 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. (1) is crucial to its performance. However, since this is a self-supervised objective, it can be incorporated without additional annotation costs.

Method	Map	Side (m)	Back (m)	Speed	DS ↑
AIM-BEV [17]	✓	19.2	0	-	45.06±1.68
Roach [11]	✓	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 vehicle, encoding vehicle speeds, and scaling to large models/datasets is crucial for performance of both PlanCNN and PlanT.

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 [68].

4.2 Explainability: Identification of Most Relevant Objects

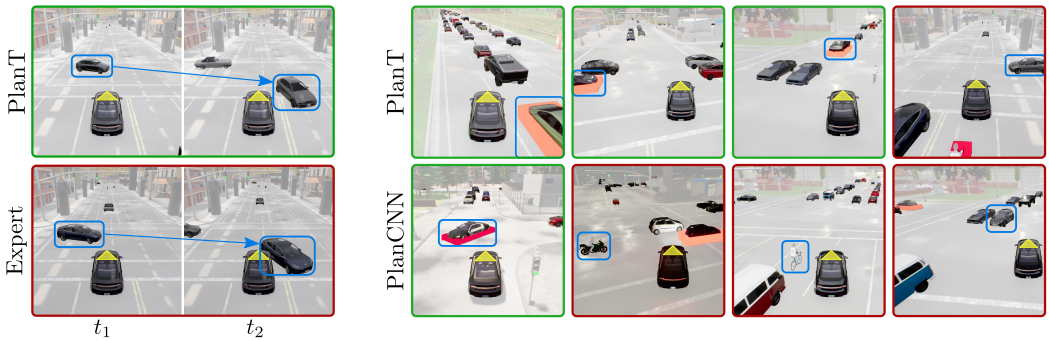
Next, we investigate the explainability of PlanT and PlanCNN by analyzing the objects in the scene that are relevant and crucial for the agent's decision. In particular, we measure the relevance of an object in terms of the learned attention for PlanT and by considering the impact that the removal of 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 [51, 50, 49]. 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 total distance. In the supplementary material, we provide additional results using GradCAM [47] adapted for PlanCNN, which we found to not work as well as the proposed masking approach.

Results. We provide results for the reasoning of the planners 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 when only a few vehicles are in the field of view or the relevant vehicle is the only one close to the ego vehicle's path. However, PlanT is also able to correctly

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, vehicles that either lead to collisions or are intuitively the most relevant in the scene are marked with a blue box.

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 from the intended route at a high speed and are therefore not as relevant for planning. 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, but nothing directly ahead, such as problems attending to the relevant object when merging lanes (examples 2 & 3). To better assess the driving performance and relevance scores we provide additional results in the supplementary video.

4.3 Limitations

Our study has several limitations. Firstly, since planning is a challenging task in itself and the capabilities of learning-based planners are not fully understood, we restrict the scope of our experiments to exclude the task of perception from sensor inputs. Integrating PlanT with a state-of-the-art perception system for object detection and tracking is a promising next step. Second, the expert driver used in our IL-based training strategy does not achieve a perfect score in our evaluation setting (Table 1) and has certain consistent failure modes (Fig. 3a, more examples can be found in the supplementary material). While a human driver may be able to achieve higher driving scores, human data collection would be time-consuming (the $3\times$ dataset used in our experiments contains around 95 hours of driving). Instead, we use the best publicly available algorithmic expert, which is a standard practice for IL on CARLA [69, 70, 34, 18]. Finally, all our experiments are conducted in simulation. Real-world scenarios are more diverse and challenging. However, CARLA is a high-fidelity simulator, and previous findings demonstrate that systems developed in simulators like CARLA can be transferred to the real world [30, 71, 72].

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. Based on our experiments, we find that it is possible to replace dense, high-dimensional feature grids including HD map information with a simple representation of the ego vehicle route without a drop in performance. 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 on the CARLA simulator. Additionally, we show that PlanT can reliably identify the most relevant object in the scene via a new metric and evaluation protocol that focus on explainability.

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] 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.

[12] 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.

[13] 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.

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

[15] E. S. Spelke and K. D. Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007. doi:https://doi.org/10.1111/j.1467-7687.2007.00569.x.

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

[17] 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.

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

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, , and A. Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *arXiv.org*, 2205.15997, 2022.

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

[27] 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.

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

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

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

[31] 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.

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

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

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

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

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

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

[38] 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.

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

[40] 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.

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

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

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

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

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

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

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

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

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

[54] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

[55] S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *ICML*, 2018.

[56] G. Liu, O. Schulte, W. Zhu, and Q. Li. Toward interpretable deep reinforcement learning with linear model u-trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018.

[57] R. M. Annasamy and K. Sycara. Towards better interpretability in deep q-networks. In *AAAI*, 2019.

[58] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. In *NeurIPS*, 2019.

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

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

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

[62] 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.

[63] 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.

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

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

[66] 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.

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

[68] 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.

[69] 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 *CVPR*, 2020.

[70] E. Ohn-Bar, A. Prakash, A. Behl, K. Chitta, and A. Geiger. Learning situational driving. In *CVPR*, 2020.

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

[72] 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.