CarFormer: Self-Driving with Learned Object-Centric Representations

Shadi Hamdan and Fatma Güney

Department of Computer Engineering, Koç University
KUIS AI Center
{shamdan17,fguney}@ku.edu.tr

Abstract. The choice of representation plays a key role in self-driving. Bird's eye view (BEV) representations have shown remarkable performance in recent years. In this paper, we propose to learn object-centric representations in BEV to distill a complex scene into more actionable information for self-driving. We first learn to place objects into slots with a slot attention model on BEV sequences. Based on these object-centric representations, we then train a transformer to learn to drive as well as reason about the future of other vehicles. We found that object-centric slot representations outperform both scene-level and object-level approaches that use the exact attributes of objects. Slot representations naturally incorporate information about objects from their spatial and temporal context such as position, heading, and speed without explicitly providing it. Our model with slots achieves an increased completion rate of the provided routes and, consequently, a higher driving score, with a lower variance across multiple runs, affirming slots as a reliable alternative in object-centric approaches. Additionally, we validate our model's performance as a world model through forecasting experiments, demonstrating its capability to predict future slot representations accurately. The code and the pre-trained models can be found at https://kuis-ai.github.io/CarFormer/.

Keywords: Intermediate Representations · Self-Driving · Object-Centric Learning

1 Introduction

The task of urban driving requires understanding the dynamics between objects in the scene. In self-driving, a scene is observed by the ego-vehicle via its sensors, typically multiple cameras. Then, for each observation step, an action is predicted and performed by the vehicle. In this paper, we propose an auto-regressive transformer-based model to reason about the dynamics of the scene while learning to drive. Starting with language processing, transformers are the de facto architecture today for sequential modeling from video generation [32, 41, 50] to robotics tasks. In language modeling, transformers typically operate on discretized inputs, or tokens, which are vastly different than high-dimensional continuous inputs such as camera images as in the case of self-driving.

There are various representation spaces used in self-driving ranging from pixels and coordinates to stixels and 3D points [22]. In recent years, efforts converged to the bird's eye view (BEV) representation [10] which provides a top-down summary of the scene with scene elements that are relevant to driving such as lanes and vehicles. It enables the agent to concentrate on relevant semantics, free of distractions. However, despite its benefits, extracting accurate BEV maps from images has proven to be a challenging problem [20, 28, 39]. Therefore, we assume access to ground-truth BEV and start by discretizing it with a VQ-VAE to provide it as input to the transformer. Although BEV provides a summary compared to six high-resolution camera images, it is still very high-dimensional. For example, most pixels belong to the road region and vehicles cover a relatively small portion of the BEV map despite being the primary cause of infractions. Our initial experiment with discretized BEV resulted in high infractions, motivating us to shift our focus to object-centric representations.

In object-centric approaches [42, 48], the scene is represented in terms of objects for better modeling of scene dynamics. In simpler robotics environments [48], this is achieved by placing objects into slots which then serve as tokens to model the interactions between objects. However, in the context of driving, learning to place objects into slots from driving sequences poses a significant challenge [15]. Previous work has addressed this challenge by representing objects with vectors containing exact object attributes, including position, size, heading, and speed [42]. In our approach, we learn to extract relevant information about objects from their spatial and temporal context in BEV sequences. This allows our model to decide how to represent objects without explicitly specifying a set of attributes that may be incomplete, or subject to variations depending on the scene.

In this paper, we propose a learned object-centric approach to self-driving with slots. In a two-stage approach, we first learn to extract slots from BEV sequences with slot attention from videos [26]. As a result of the first step, ideally, each object is placed into a slot that contains relevant information about the object. Each slot is naturally self-contained as it is responsible for reconstructing the object that it binds to. Slot extraction is trained with a self-supervised objective and, therefore presents a more feasible and scalable alternative to exact object attributes. Moreover, it holds promise for generalization to diverse objects beyond vehicles. In the experiments, we show that slots can infer necessary information for driving such as the speed and the orientation of the vehicles. We analyze the factors that affect slot extraction and behavior learning with slots and find that increasing the number of slots and enlarging small objects improve performance significantly.

In the second step, we use a transformer to jointly learn the driving and the scene dynamics based on the extracted slot representations. We propose to replace the causal attention mechanism typically used in autoregressive transformers with block attention to allow interaction between all objects and the route for better modeling of dynamics in the scene. For supervision, we apply loss on both continuous actions predicted using a GRU as in PlanT [42] and on

quantized actions predicted by the transformer autoregressively in multiple steps. We perform extensive evaluations in terms of design choices including block attention and the choice of action head. In comparison to exact object attributes, slots result in a better completion rate of provided routes, leading to a higher driving score. Importantly, variation across multiple runs is also significantly lower with slots, which is a sign of robustness to variations in the scene during online evaluation. In addition to improved driving performance, we qualitatively demonstrate that our model indeed learns dynamics with accurate predictions of slot representations in future steps. Our contributions are, in summary:

- A learned, self-supervised, object-centric representation for self-driving based on slot attention, that contains the information necessary for driving such as speed and orientation of vehicles without explicitly providing them.
- CarFormer, an autoregressive transformer, that can both drive and act as a world model, predicting future states.
- State-of-the-art performance in the privileged setting of Longest6 benchmark, outperforming exact object-level attributes.

2 Related Work

We first provide an overview of representation spaces that are commonly used in self-driving with a special focus on BEV. We then briefly summarize the progress in self-supervised object-centric methods. Finally, we compare our approach to the recent sequence modeling approaches to control problems in robotics.

Representation Spaces in Self-Driving: Starting with hand-designed affordances [7, 43], various representation types have been proposed for self-driving. One common representation is semantic segmentation, initially in 2D [33, 34, 44, 55], sometimes coupled with object detection for efficiency [4], and more recently in BEV [8,9,19,21,53]. Another representation space is the coordinate space that is commonly used in trajectory prediction [6]. A notable work by Wang et al. uses a 2D detector to estimate the 3D properties of objects to render them in BEV [47]. In addition to position, PlanT [42] also includes other driving-related information such as heading and speed in its representation of each object through an attribute vector. A representation based on object coordinates enables methods to concentrate on object-to-object relationships. However, it often lacks rich semantic information and may not adequately capture the varying spatiotemporal contexts of objects. Our goal is to address these limitations by integrating learned object-centric representations.

Self-Supervised Object-Centric Representations: In this work, we build on the progress in object-centric learning where the goal is to decompose the scene into objects. A common way of achieving this goal is to integrate inductive biases about objects into the architecture, typically in an auto-encoding paradigm. Beginning with slot attention [31], these techniques reconstruct the input with a set of bottlenecks in the latent space called *slots*. Each slot is expected to bind to an object region with similar visual cues. In this work, we

adopt the methodology proposed by SAVi [26], which extends slot attention to video by incorporating temporal dynamics.

The progress in this domain started with synthetic images [24, 25], and has since shifted towards in-the-wild images [17, 29] and real-world videos [27, 36, 38, 49, 52]. However, the existing methods struggle due to the complexity of unconstrained scenarios and resort to reconstructing different modalities such as flow [26], depth [15] or motion segmentation masks [2,3]. While recent studies [1, 45] show promising results by performing reconstruction in the feature space of self-supervised models [5], extracting slots from complex driving sequences remains a significant challenge. In this work, we assume the availability of a BEV representation of the scene through time as BEV resembles synthetic sequences where these methods perform reliably.

Transformers for Sequential Modeling: Transformers are commonly applied to sequential prediction tasks in vision, such as video generation [32,35,41, 51]. Previous work in robotics formulates Reinforcement Learning as a sequence modeling problem [11,18,23,54]. However, these methods are typically evaluated on tasks where the state space is low-dimensional or can be straightforwardly encoded into a single vector. For instance, in Decision Transformer [11], the state is assumed to be adequately represented with a single token, which is encoded into a single vector with a CNN in the case of visual inputs. Trajectory Transformer [23] processes continuous states and actions by discretizing every dimension separately. While this works for low-dimensional action spaces, it is infeasible for the high-dimensional state space of self-driving. One potential solution that we explore is to employ a VQ-VAE to quantize the input image into a lower dimensional discrete representation [16].

We do not constrain the state space to being discretized. Instead, we employ a hybrid modality for the inputs where some features are continuous while others are discrete. Although we utilize continuous inputs and non-causal *block* attention, we maintain the autoregressive capability to generate future rollouts and reason on them, a crucial aspect of transformer-based RL approaches [23]. Non-causal *block* attention is similar to the non-causal transformer decoder introduced in [30], but we do not limit the block to appear solely at the beginning of the sequence.

3 Background on Slot Extraction

Given the BEV representation of the scene in the last T time steps, we first use a frozen object-centric model to extract the objects into slots. For extracting slots, we build on slot attention for videos, SAVi [26]. Here, we provide a brief overview of SAVi for completeness and also, to introduce the notation with slots. Please see [26] for details of SAVi.

As a reconstruction-based method, SAVi follows an auto-encoding paradigm. Given a sequence of RGB frames $\mathbf{x}_{t-T:t}$ with T time steps for context, a CNN-based encoder is used to process each frame $\mathbf{x}_i$ with positional encoding. The output of the encoder is flattened into a set of vectors $\mathbf{h}_{t-T:t}$. SAVi first initializes

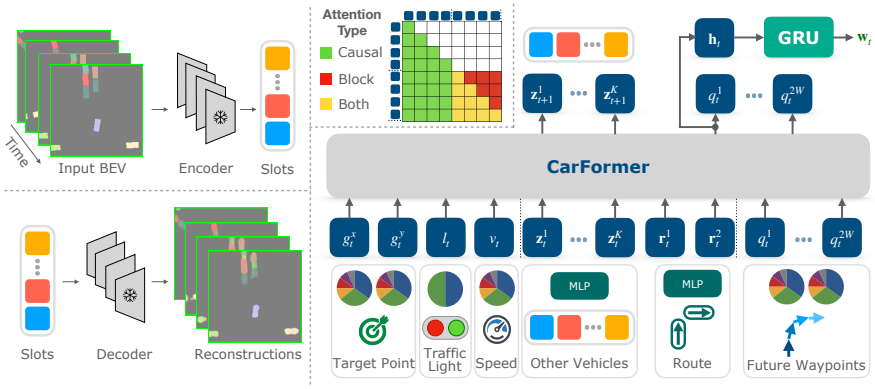

Fig. 1: Overview of CarFormer. Given a trajectory τ_t consisting of discrete and continuous inputs, we first embed these tokens to the same hidden dimension H. For scalar inputs such as target point (g_t^x, g_t^y), traffic light flag l_t, speed v_t, and waypoints q_t^i, we discretize them, using k-means if not discrete, before we look them up from an embedding matrix. For continuous inputs like the K slot features $\{\mathbf{z}_t^i\}_{i=1}^K$ and the desired route $\mathbf{r}_t^1, \mathbf{r}_t^2$, we project them using an MLP. Conditioned on this context, CarFormer learns to jointly predict future slot features $\{\mathbf{z}_{t+1}^i\}_{i=1}^K$ and the waypoints autoregressively using the backbone $(\mathbf{q}_t)$ as well as the GRU head $(\mathbf{w}_t)$. The K slot features $\{\mathbf{z}_t^i\}_{i=1}^K$ are extracted from a pre-trained, frozen, SAVi model, shown on the left. The K slot features, along with the desired route, are considered a block, and block attention is applied in the attention layers as shown on the top.

K slot vectors $\tilde{\mathcal{Z}}_{t-T} = \{\mathbf{z}_{t-T}^i\}_{i=1}^K$. The initial set of slot vectors $\tilde{\mathcal{Z}}_i$ is then updated with Slot Attention (SA) [31] based on the visual features $\mathbf{h}_i$ from the encoder for each time step i within the interval, resulting in the updated set of slot vectors $\mathcal{Z}_i$:

$$\mathcal{Z}_i = f_{SA}\left(\tilde{\mathcal{Z}}_i, \mathbf{h}_i\right) \tag{1}$$

To ensure temporally consistent slot representations, the subsequent slots are initialized based on the slot representation of the previous time step: $\mathcal{Z}_{i+1} = f_{pred}\left(\mathcal{Z}_i\right)$. In SAVi, slots are decoded into RGB predictions, and an alpha mask per slot for computing the reconstruction loss. We use the latent slot features $\mathcal{Z}_t$ to describe time step t in our model, i.e., without decoding.

4 Methodology

We introduce CarFormer for learning to drive in the urban environment of CARLA [14]. Urban driving presents complexity due to the interactions between the ego-vehicle and other vehicles. Our goal is to learn driving behavior by capturing scene dynamics through slot representations. We formulate the behavior learning as a sequence modeling problem, as illustrated in Fig. 1. This sequence comprises tokens representing the goal, state, and action. We first define representations for each aspect before detailing the model architecture.

4.1 Goal and State Representation

We encode the route to follow and the state of the world in terms of tokens, which can be continuous or discrete, and feed them into our model. Specifically, we provide the model with the next target point in the route (g_t^x, g_t^y), a flag signifying whether the ego vehicle is affected by a red traffic light $l_t \in \{0, 1\}$, and the current speed of the ego vehicle $v_t \in \mathbb{R}_0^+$. For each of these attributes, we apply k-means clustering and quantize them into k_{attr} bins. For scene representation, we initially consider a scene-level representation by directly encoding the BEV map, and then we explore object-level representations.

Scene-Level Representation: The input consists of a BEV representation of the scene at time t, denoted as $\mathbf{B}_t \in 0, 1^{192\times192\times8}$, centered around the ego-vehicle at time t (i.e. we use an ego-centric coordinate frame at each time-step). Each of the 8 channels represents a binary map corresponding to a semantic class, such as road and vehicles [53]. Following common practice [51], we use a VQ-VAE [37] to encode $\mathbf{B}$ into a grid of discrete integers: $\mathbf{b}_t \in \{1, \ldots, C\}^{12\times12}$ where C denotes the codebook size. We then flatten this grid to obtain a representation of the scene as a set of discrete tokens. Despite successful reconstructions, our experiments show that learning successful driving behavior on top of these discretized tokens is challenging (see Table 1).

Object-Level Representations: An alternative approach to representing the entire scene as a rasterized BEV is to represent individual objects within the scene. In PlanT [42], both vehicles and the desired route are represented as 6-dimensional vectors representing essential information such as size, position, orientation, and speed. Following PlanT, we initially train our model using the exact attributes of objects to represent the scene. This allows us to attribute improvements directly to the proposed object-centric representation with slots rather than our modifications to the transformer, such as block attention.

In this paper, we propose an alternative object-level representation based on slots. Building on advancements in self-supervised object-centric representations, we explore slots as a more natural way of representing objects compared to exact object attributes. With slots, objects can be implicitly represented with relevant information from the spatio-temporal context of the object. Given the BEV representation of the scene in previous T time steps, denoted as $\mathbf{B}_{t-T:t}$, we employ SAVi [26] to extract objects into K slots, $\{\mathbf{z}_t^i\}_{i=1}^K$ where each $\mathbf{z}_t^i \in \mathbb{R}^{1\times d}$ is a d-dimensional slot vector corresponding to an object in the scene. As object-level representations do not include any information on the desired route, we also provide the model with the desired route represented by two vectors $\mathbf{r}_t^1, \mathbf{r}_t^2 \in \mathbb{R}^6$ following PlanT [42].

4.2 Action Representation

Following common practice in self-driving [12, 19, 42], we predict waypoints that are then used to calculate the corresponding control consisting of throttle, brake, and steering angle. We predict waypoints in two ways:

GRU: One way is to use a small GRU [13] as in PlanT [42] The GRU is fed with the last latent vector from the backbone, concatenated with a flag representing the traffic light status. With the GRU head, we sequentially predict W future waypoints $\mathbf{w}_t = \{(w_{x,i}, w_{y,i})\}_{i=t+1}^{t+W}$ one by one in an autoregressive manner.

Quantization: Another way is to use our autoregressive transformer backbone and treat waypoint prediction as a next-word problem, a common strategy in language modeling approaches for RL [11,23,46]. To implement this, we quantize the 2D waypoints by using k-means clustering on each dimension separately. As a result, W waypoints are represented by a vector $\mathbf{q}_t \in \{1, \ldots, k_q\}^{2W \times 1}$, and then each dimension is predicted sequentially, conditioned on the previous predictions.

4.3 CarFormer

Our goal is to learn self-driving in urban environments while jointly reasoning about scene dynamics as a sequence prediction task. At each time step, we represent the state of the world with a set of tokens as previously defined. We define a trajectory τ_t at time step t as follows:

$$\tau_t = \{g_t^x, g_t^y, l_t, v_t, \mathbf{z}_t^1, \ldots, \mathbf{z}_t^K, \mathbf{r}_t^1, \mathbf{r}_t^2, q_t^1, \ldots, q_t^{2W}\} \tag{2}$$

where g_t^x, g_t^y denote the target point, l_t the traffic light, v_t the speed, $\{\mathbf{z}_t^i\}_{i=1}^K$ object-level slot representations, and $\mathbf{r}_t^1, \mathbf{r}_t^2$ two attribute vectors representing the desired route. In PlanT style representation, slot vectors are replaced by the object attributes. In the case of scene-level representation, object-level slot representations and the desired route are replaced by the discrete tokens from the VQ-VAE, represented as $\{b_t^i\}_{i=1}^C$.

Encoding: For each input in the trajectory, we handle discrete inputs, such as quantized target points and traffic light status, by performing a lookup from an embedding matrix. In the case of continuous attributes, like slot vectors, we project the vectors into the desired dimensionality using an MLP, as illustrated in Fig. 1. Specifically, for each discrete attribute, we initialize a $k_{attr} \times H$ embedding matrix, where k_{attr} represents the number of possible values of the attribute after discretization, and H denotes the hidden dimension of the backbone. Conversely, for continuous attributes, we utilize an MLP to project the vectors into $\mathbb{R}^H$.

Architecture: The backbone of the CarFormer is an autoregressive transformer decoder adapted from the architecture of GPT-2 [40]. We modify the embedding layer to accommodate both continuous and discrete inputs simultaneously, enabling the incorporation of continuous representations such as slots while keeping other inputs like speed and traffic light discrete. Furthermore, we adjust the attention mechanism, which is causal in transformer decoders, to allow for cross-attention between certain blocks of the input. These blocks can be considered as a single unit, despite being composed of multiple tokens, such as the slot representations. To achieve this, we replace the triangular causal attention mask with a block triangular mask, enabling cross-attention within the block as shown on top in Fig. 1. We use this attention mechanism throughout our experiments and evaluate its impact in Table 2.

4.4 Training

While our formulation can be extended to RL with rewards and multi-step predictions, currently, it corresponds to imitation learning with a single-step policy to predict action based on context. We train the model using imitation learning to learn to predict both the continuous waypoints $\mathbf{w}_t$ and their quantized form, $\mathbf{q}_t$ given the context as shown in (2). We supervise both the GRU head, responsible for predicting continuous waypoints, and the language modeling head, responsible for predicting the quantized waypoints. To achieve this, we use the following loss functions:

$$\begin{aligned} \mathcal{L}_{wp} &= \mathcal{L}_{\text{GRU}} + \mathcal{L}_{\text{LM}} \\ \mathcal{L}_{\text{GRU}} &= \sum_{i=1}^{W} |\mathbf{w}_t^i - \hat{\mathbf{w}}_t^i| \\ \mathcal{L}_{\text{LM}} &= \sum_{i=1}^{W} \sum_{j=1}^{2} \text{CE}(\mathbf{q}_t^{i,j}, \hat{\mathbf{q}}_t^{i,j}) \end{aligned} \tag{3}$$

Auxiliary Forecasting: Similar to PlanT [42], we additionally train the model to predict future scene representations jointly with action. In the case of the discretized scene representation using the VQ-VAE, we calculate the cross-entropy between the predicted logits and the ground truth future representation:

$$\mathcal{L}_{\text{scene}} = \sum_{i=1}^{C} \text{CE}(\mathbf{b}_{t+f}^i, \hat{\mathbf{b}}_{t+f}^i) \tag{4}$$

where f denotes the time horizon into the future for which we make predictions.

In the case of continuous scene representations, such as in object-level vectors or slot representations, we instead use the mean squared error between the representations:

$$\mathcal{L}_{\text{object}} = \sum_{i=1}^{K} \|\mathbf{z}_{t+f}^i, \hat{\mathbf{z}}_{t+f}^i\| \tag{5}$$

The final loss is a weighted sum of the losses on action (3) and forecasting:

$$\mathcal{L}_{wp} + \alpha \ \mathcal{L}_{\text{forecast}} \tag{6}$$

where $\mathcal{L}_{\text{forecast}}$ corresponds to $\mathcal{L}_{\text{scene}}$ (4) in case of scene-level and $\mathcal{L}_{\text{object}}$ (5) in case of object-level. We experimentally set α, the weight of forecasting, to 40.

5 Experiments

5.1 Experimental Setup

CARLA Setting: We collect training data using the setup introduced in TransFuser [12] on CARLA version 9.10.1., as also in PlanT [42]. As PlanT [42] shows

that scaling the dataset size by collecting the data multiple times with different seeds leads to performance improvements, we adopt the 3× setting, i.e. collecting the data three times, as our default configuration. Our evaluation is conducted on the Longest6 benchmark, as proposed in TransFuser [12], comprising the six longest routes across six different towns. Further details on the expert driver and data collection routes can be found in TransFuser [12].

Data Filtering: During data collection, we filter out expert runs that exhibit problematic behavior, typically occurring with specific seeds. We perform filtering by repeating runs where the expert achieves a driving score of less than 50% with a different seed, ensuring consistency in dataset size. The filtered instances commonly involve vehicles spawned in orientations hindering movement, leading to expert timeouts, or scenarios failing to trigger, causing the expert to remain immobilized until timing out.

Evaluation Setting: During online evaluations, unless stated otherwise, we employ the model with slots as input, $t + 4$ as the target for future prediction, and the GRU head for waypoint prediction. Additionally, we implement creeping to prevent the agent from becoming stuck.

Metrics: When assessing driving performance, we report the metrics used in the CARLA leaderboard [14], namely *Route Completion (RC)*, *Infraction Score (IS)*, and *Driving Score (DS)* as the combination of the two. For evaluating the predicted dynamics, we report the foreground version of the *Adjusted Rand Index (ARI)* and mean *Intersection over Union (mIoU)*. These metrics are computed by comparing predicted slot masks to target slot masks at a future time step.

Baselines: Since we operate in the privileged setting, we benchmark against other privileged baselines. Specifically, we compare with AIM-BEV introduced in [19] and ROACH [53] which use ground truth BEV as input. Additionally, we compare to PlanT [42], which incorporates ground truth object-level vehicles and routes as input. We obtain the results for AIM-BEV and ROACH directly from [42] and evaluate the publicly available PlanT-Medium checkpoint, trained on the 3× dataset on Longest6.

Implementation Details: Our dataset is divided into a training set (94% of the data), and validation and test sets (3% each). We train for 100 epochs on the training set and select the checkpoint with the best validation loss for online evaluation. We set the hidden dimension of the transformer as $H = 768$ and the number of layers to 6. In k-means, we use $k_g = 32$ for the target point, $k_v = 14$ for the speed, and $k_q = 48$ for the quantized waypoints. For the target point and waypoints, we allocate half of the dimension for x and the other half for y. In SAVi, we set the number of context frames T to 2 and the slot dimensionality d to 128 and train it from scratch on a subset of our training split. Additional details can be found in Supplementary.

Model	Representation	DS↑	IS↑	RC↑
CarFormer	BEV	17.07±3.78	0.30±0.04	59.54±4.34
AIM-BEV* [19]		45.06±1.68	0.55±0.01	78.31±1.12
ROACH* [53]		55.27±1.43	0.62±0.02	88.16±1.52
CarFormer	Attributes	71.53±3.52	0.78±0.06	90.01±1.60
PlanT (Rep.) [42]		73.36±2.97	**0.84±0.01**	87.03±3.91
CarFormer	Slots	**74.89±1.44**	0.79±0.02	**92.90±1.28**

Table 1: Comparison on Longest6. In the top part, we report the scene-level comparison where CarFormer significantly falls behind the other two approaches. In the middle, we report results using exact, object-level attributes, where CarFormer achieves a driving score within statistical significance of PlanT. With object-level slot representations shown in the bottom, CarFormer outperforms all scene-level approaches and even surpasses object-level approaches based on explicit object attributes. We report mean±std over 3 different runs. We report the results of PlanT reproduced (Rep.) using their official code. *Results reported in PlanT [42].

5.2 Quantitative Results

Comparison: We present the results of online evaluations on the Longest6 benchmark in Table 1. The table is divided into three based on the type of representation: scene-level representation at the top, followed by exact object-level attributes, and object-level slot representations at the bottom. In scene-level representations, CarFormer lags behind another imitation learning approach AIM-BEV [19], and an RL approach, ROACH [53]. Despite accurately reconstructing input BEV with a VQ-VAE, the model cannot focus on objects, as evidenced by the significantly lower infraction score (IS).

Compared to scene-level representation with VQ-VAE, we observe significant performance improvements with object-level representations. CarFormer with slots outperforms PlanT in RC despite a lower IS due to covering a longer distance, resulting in a higher mean DS with only half the variance compared to PlanT (Table 1). This achievement is particularly noteworthy for two reasons: First, the slots model (bottom row) achieves this solely from BEV. While PlanT and CarFormer with attributes have access to exact agent locations, the slots model learns to accurately place agents in slots. Second, the significantly lower variance with slots demonstrates increased stability across runs, affirming slots as a more reliable alternative to attribute vectors. Note that the improved performance of our model cannot be attributed to architectural changes, as CarFormer with attributes performs worse than PlanT with higher variance.

Ablation Study: In Table 2, we perform an ablation study to evaluate our design choices when training and evaluating our model with slots as input. Specifically, we evaluate the effect of removing block attention, forecasting slots, and creeping when the vehicle is stuck during online evaluation. In the complete version of our model, we compare the results when using action predictions from

Action	BA	Forecasting	Creeping	DS↑	IS↑	RC↑
GRU	✓	✓	✓	**74.89±1.44**	0.79±0.02	**92.90±1.28**
Q	✓	✓	✓	66.87±0.96	0.74±0.01	87.12±0.89
GRU	×	✓	✓	70.42±2.68	0.78±0.02	88.19±2.00
GRU	✓	×	✓	57.25±2.89	0.63±0.03	89.54±1.48
GRU	✓	✓	×	69.16±2.20	**0.83±0.01**	80.52±1.88

Table 2: Ablation Study. We first compare predicting continuous actions with the GRU to predicting quantized actions with the transformer (Q). We also ablate the effect of removing Block Attention (BA) and slot forecasting and creeping. We present the results as the mean ± std over 3 different runs on Longest6.

either the GRU or the transformer itself with quantization (Q). We observed a significant improvement in results with the GRU and consequently adopted it for the remaining experiments.

The negative effect of removing creeping can be seen in a notably lower route completion score. Without creeping, the agent suffers from more frequent instances of getting stuck, leading to a decreased completion rate of assigned routes. This cautious behavior leads to a slightly higher IS but a lower overall DS. With BA, the agent can better model relationships between all objects in the scene due to the bi-directional attention within a block. As shown in the third row, removing BA results in increased infractions and a lower RC score. Adding forecasting significantly improves driving performance. The effect of forecasting is most visible in infraction scores, where removing it leads to a 0.17 decrease in IS ($0.78 \rightarrow 0.63$). This shows the importance of formulating the driving task jointly with forecasting, allowing the agent to anticipate the intentions of other vehicles and act accordingly.

Slot Representations: When examining the performance of SAVi on BEV sequences, we identified two primary factors contributing to failure in slot extraction. Firstly, a low number of slots leads to missing vehicles as all slots become filled, particularly in crowded urban driving scenes. As demonstrated in Table 3, the performance improvement from 7 to 30 slots clearly highlights the advantage of setting the number of slots high enough to capture majority of vehicles in the driving environment. However, adding more slots increases the computational load of training SAVi. To address this, we developed a lighter decoder to maintain efficiency while accommodating more slots. We provide an ablation of the light decoder in Supplementary.

Increasing the number of slots leads to significant performance improvements, particularly in terms of infractions. Upon investigating the infractions with the lower number of slots, we discovered that most of them stemmed from small vehicles such as motorbikes or bicycles. Small vehicles, covering a relatively small area on the final BEV map, were often missed while decoding slots due to the small cost paid in terms of reconstruction loss. To test our hypothesis, we enlarged these small vehicles to increase their likelihood of being assigned to a

slot. As shown at the bottom part of Table 3, enlarging small vehicles notably enhances the performance, particularly with a small number of slots. The best performance is obtained with 30 slots and enlarging small vehicles.

Forecasting Future Slots: So far, we evaluated the performance of our model as a policy function to predict action. Additionally, our model can also predict future states of objects by learning to match the slot representations of SAVi in future time steps via the loss function defined in (5). Therefore, CarFormer can be evaluated as a world model for predicting future states in addition to action. Similar to prior work on visual dynamics models [48], we evaluate our model's performance in directly predicting future slot representations 1 or 4 timesteps ahead in Table 4. To be able to use object discovery metrics ARI and mIoU, we decode the predicted slot representations using the frozen decoder of SAVi.

We report the performance of directly copying input as a sanity check (Input-Copy) and the performance of SAVi as an upper bound since SAVi has access to the current frame. Input-Copy works well for predicting the near future but degrades while predicting timestep $t+4$ due to objects moving away from their initial position. While there is a drop in our model's performance in predicting timestep $t+4$ as well, its predictions are more accurate in all metrics. Our model can learn the dynamics of objects in the scene and change their future position and orientation with respect to input.

5.3 Qualitative Results

In Fig. 2, we visualize predictions of our model (third column) in comparison to ground truth (first column) and SAVi reconstructions (second column) which we use to supervise our model for predicting future slot representations. In the first three columns, we display the initial locations of the vehicles in dark grey and the final locations that we aim to predict in light grey. In the last column, we overlay the final positions in all three columns, each represented in a different channel: ground truth in red, SAVi in green, and CarFormer in blue. This allows us to clearly identify various types of errors for our model in a specific color such as false negatives in blue and false positives in yellow.

#Slots	Enlarged	DS↑	IS↑	RC↑
7		48.17±8.61	0.56±0.08	86.70±5.21
14	×	49.34±5.66	0.54±0.07	92.33±2.43
30		71.48±5.25	0.75±0.06	93.00±0.23
7		62.93±6.78	0.73±0.07	80.20±1.87
14	✓	69.75±8.03	0.78±0.04	86.17±2.79
30		**74.89±1.44**	**0.79±0.02**	**92.90±1.28**

Table 3: Effect of Slot Extraction on Driving. We perform experiments to show the effect of enlarging small vehicles in order to overcome perception issues as well as varying the number of slots. We report mean±std over 3 different runs on Longest6.

Method	$T = t + 1$		$T = t + 4$	
	ARI↑	**mIoU↑**	**ARI↑**	**mIoU↑**
Input-Copy	0.641	0.561	0.412	0.375
CarFormer	0.795	0.702	0.540	0.454
SAVi	0.924	0.874	0.924	0.874

Table 4: Forecasting Results. We evaluate CarFormer's ability to predict future slots at time $T = T + 1$ and $T = T + 4$ that are decoded with the frozen SAVi decoder.

Proper Dynamics: As illustrated in examples a, c, d, and e, our model effectively learns visual dynamics between objects. In a, our model recognizes that the vehicle in the bottom left is moving backward (downward), and accurately predicts its absence in the future. In the same example, our model correctly infers that vehicles on the right are heading forward at a slower speed than the ego vehicle, and thus accurately predicts their future locations. In d, our model accurately identifies that all vehicles are heading forward, with those on the top-left moving slightly faster than the ego vehicle, thus they end up slightly further up. These examples demonstrate that our model can deduce vehicle speeds from slot representations, without explicit information on heading direction or speed. Its capability extends beyond simply associating left/right lanes with heading direction, as can be seen from a and d. In case of a lane change (e) or a turn (c), our model accurately predicts the change in its viewpoint.

Problems in Forecasting: We encounter three types of issues while forecasting slots. The first one occurs due to the perception errors by SAVi during slot extraction. Since we rely on slot representations extracted by SAVi both as input and also for supervision while forecasting slots, our model's performance is constrained by SAVi's performance. This can be observed from errors caused by blurry predictions of SAVi in turning scenarios in e and c or from failing to locate cars in crowded scenes in b. The second type of problem involves False Positives, resulting from hallucinated vehicles, as indicated by the blue color in the last column of e, c, and f. Lastly, our model struggles to predict complex dynamics when multiple vehicles are moving at different speeds as shown in c.

6 Conclusion and Future Work

In this paper, we introduced CarFormer as the first approach to self-driving with object-level slot representations. We demonstrated that reasoning with slots not only improves the driving score but also provides robustness across variations in multiple online evaluations. We trained and validated the performance of CarFormer both as a policy to predict action and as a visual dynamics model for predicting future states of objects. Unlike PlanT, which utilizes a transformer encoder to process a single time step, we employed an autoregressive transformer decoder in CarFormer. This design has the potential to be extended to multi-step

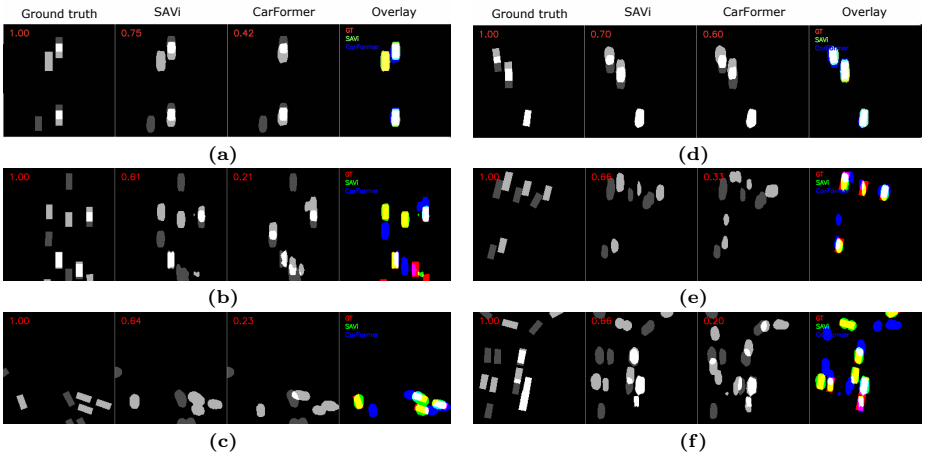

Fig. 2: Visualization of Slot Forecasting Results. Each sub-figure shows an example of input (dark grey)-output (light grey) objects in the first column, SAVi reconstructions in the second column, and our model's predictions in the third column. The top left corner of each column shows the mIoU compared to the ground truth. For comparison, we overlay the three in the last column where the red channel (R) is the ground-truth location, the green channel (G) is SAVi reconstruction, and the blue channel (B) is our prediction. In the case of perfect alignment between the three, we see the vehicles in white, and different errors for our model can be seen in a unique color such as yellow (R+G) indicating misses and blue indicating false positives (B).

reasoning with reward/return tokens, as demonstrated in robotics tasks [11, 23]. In comparison to robotics tasks, self-driving has a more complex state representation due to intricate dynamics between objects in addition to well-known challenges in appearance, especially when extracting information from cameras.

We currently assume access to the ground truth BEV maps in our model. Despite significant progress in learning BEV representations in recent years, BEV perception still lacks the accuracy needed to extract slots from it in urban driving scenes. Instead of a two-stage approach by first estimating BEV and then extracting slots from it, a more direct approach of extracting slots in BEV might be preferable, both in terms of efficiency and avoiding cascading errors. With the advancements in slot extraction from real-world videos, any object that can be placed into a slot can be part of the reasoning in our model.

Acknowledgements: We thank Barış Akgün, Deniz Yuret, and members of AVG at Koç University for discussions and proof-reading, Hongyang Li and Li Chen for helpful suggestions during the rebuttal, and Katrin Renz for sharing details of PlanT. This project is co-funded by the KUIS AI Center and the European Union (ERC, ENSURE, 101116486). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Aydemir, G., Xie, W., Güney, F.: Self-supervised Object-centric Learning for Videos. In: NeurIPS (2023)
2. Bao, Z., Tokmakov, P., Jabri, A., Wang, Y.X., Gaidon, A., Hebert, M.: Discovering objects that can move. In: CVPR (2022)
3. Bao, Z., Tokmakov, P., Wang, Y.X., Gaidon, A., Hebert, M.: Object discovery from motion-guided tokens. In: CVPR (2023)
4. Behl, A., Chitta, K., Prakash, A., Ohn-Bar, E., Geiger, A.: Label efficient visual abstractions for autonomous driving. In: IROS (2020)
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV (2021)
6. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3D tracking and forecasting with rich maps. In: CVPR (2019)
7. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: ICCV (2015)
8. Chen, D., Krähenbühl, P.: Learning from all vehicles. In: CVPR (2022)
9. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. In: CORL (2019)
10. Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., Li, H.: End-to-end autonomous driving: Challenges and frontiers (2024), https://arxiv.org/abs/2306.16927
11. Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. In: NeurIPS (2021)
12. Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., Geiger, A.: Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. IEEE TPAMI (2023)
13. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
14. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: CORL (2017)
15. Elsayed, G.F., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M.C., Kipf, T.: SAVi++: Towards end-to-end object-centric learning from real-world videos. In: NeurIPS (2022)
16. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: CVPR (2021)
17. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV (2010)
18. Furuta, H., Matsuo, Y., Gu, S.S.: Generalized decision transformer for offline hindsight information matching. In: ICLR (2022)
19. Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., Geiger, A.: King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In: ECCV (2022)
20. Harley, A.W., Fang, Z., Li, J., Ambrus, R., Fragkiadaki, K.: Simple-BEV: What really matters for multi-sensor bev perception? In: ICRA (2023)
21. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., Lu, L., Jia, X., Liu, Q., Dai, J., Qiao, Y., Li, H.: Planning-oriented autonomous driving. In: CVPR (2023)

22. Janai, J., Güney, F., Behl, A., Geiger, A.: Computer vision for autonomous vehicles: Problems, datasets and state of the art. Foundations and Trends® in Computer Graphics and Vision **12**(1–3), 1–308 (2020). https://doi.org/10.1561/0600000079, http://dx.doi.org/10.1561/0600000079
23. Janner, M., Li, Q., Levine, S.: Offline reinforcement learning as one big sequence modeling problem. In: NeurIPS (2021)
24. Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: CVPR (2017)
25. Karazija, L., Laina, I., Rupprecht, C.: Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. ARXIV **2111.10265** (2021)
26. Kipf, T., Elsayed, G.F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., Greff, K.: Conditional Object-Centric Learning from Video. In: ICLR (2022)
27. Li, F., Kim, T., Humayun, A., Tsai, D., Rehg, J.M.: Video segmentation by tracking many figure-ground segments. In: CVPR (2013)
28. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: ECCV (2022)
29. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
30. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.M.: Generating wikipedia by summarizing long sequences. In: ICLR (2018)
31. Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., Kipf, T.: Object-centric learning with slot attention. In: NeurIPS (2020)
32. Micheli, V., Alonso, E., Fleuret, F.: Transformers are sample-efficient world models. In: ICLR (2023)
33. Mousavian, A., Fiser, M., Davidson, J., Kosecka, J., Toshev, A.: Visual representations for semantic target driven navigation. In: ICRA (2019)
34. Müller, M., Dosovitskiy, A., Ghanem, B., Koltun, V.: Driving policy transfer via modularity and abstraction. In: CORL (2018)
35. Nash, C., Carreira, J., Walker, J., Barr, I., Jaegle, A., Malinowski, M., Battaglia, P.: Transframer: Arbitrary frame prediction with generative models. ARXIV **2203.09494** (2022)
36. Ochs, P., Malik, J., Brox, T.: Segmentation of moving objects by long term video analysis. IEEE TPAMI (2013)
37. van den Oord, A., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: NeurIPS (2017)
38. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: CVPR (2016)
39. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: ECCV (2020)
40. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
41. Ren, X., Wang, X.: Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In: CVPR (2022)
42. Renz, K., Chitta, K., Mercea, O.B., Koepke, A.S., Akata, Z., Geiger, A.: PlanT: explainable planning transformers via object-level representations. In: CORL (2022)

43. Sauer, A., Savinov, N., Geiger, A.: Conditional affordance learning for driving in urban environments. In: CORL (2018)
44. Sax, A., Emi, B., Zamir, A.R., Guibas, L.J., Savarese, S., Malik, J.: Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. In: CORL (2019)
45. Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.J., He, T., Zhang, Z., Schölkopf, B., Brox, T., et al.: Bridging the gap to real-world object-centric learning. In: ICLR (2023)
46. Shafiullah, N.M.M., Cui, Z.J., Altanzaya, A., Pinto, L.: Behavior transformers: Cloning k modes with one stone. In: NeurIPS (2022)
47. Wang, D., Devin, C., Cai, Q.Z., Krähenbühl, P., Darrell, T.: Monocular plan view networks for autonomous driving. In: IROS (2019)
48. Wu, Z., Dvornik, N., Greff, K., Kipf, T., Garg, A.: SlotFormer: unsupervised visual dynamics simulation with object-centric models. In: ICLR (2023)
49. Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., Huang, T.: Youtube-vos: A large-scale video object segmentation benchmark. In: ECCV (2018)
50. Yan, W., Zhang, Y., Abbeel, P., Srinivas, A.: Videogpt: Video generation using VQ-VAE and transformers. CoRR **abs/2104.10157** (2021), `https://arxiv.org/abs/2104.10157`
51. Yan, W., Zhang, Y., Abbeel, P., Srinivas, A.: VideoGPT: video generation using VQ-VAE and transformers. ARXIV **2104.10157** (2021)
52. Yang, L., Fan, Y., Xu, N.: Video instance segmentation. In: CVPR (2019)
53. Zhang, Z., Liniger, A., Dai, D., Yu, F., Van Gool, L.: End-to-end urban driving by imitating a reinforcement learning coach. In: ICCV (2021)
54. Zheng, Q., Zhang, A., Grover, A.: Online decision transformer. In: ICML (2022)
55. Zhou, B., Krähenbühl, P., Koltun, V.: Does computer vision matter for action? Science Robotics **4** (2019)