CarFormer: Self-Driving with Learned Object-Centric Representations

Supplementary Material

Abstract. In this Supplementary, we first provide the **details** about the architecture (Section 1) and training (Section 2) for reproducibility. We then provide **additional quantitative results** in Section 3 on ablating our decision to use a lighter architecture in slot extraction and on the effect of enlarging slots and block attention in terms of forecasting performance. We also report the **inference time and the number of parameters** of our model at the end of Section 3. We **visually compare** slot reconstructions between different versions of SAVi as well as forecasting results by our model to ground truth and SAVi in Section 4. Finally, we provide a road map for **future work** to autoregressively generate rollouts with block attention in Section 5. Please also see the **video** in the zip for a visualization of our model's driving performance.

1 Architectural Details

Slot Extraction: To extract slots, we build on SAVi [32] based on its implementation in [51]. Additionally, we process the input BEV representations by assigning a different color to each vehicle, which we found to help in slot extraction. To color the vehicles, we sample a random color per vehicle from a set of 14 colors and fix this color by the vehicle ID. To use enlargement, we enlarge any vehicles that are less than $4.9m \times 2.12m$ to $4.9m \times 2.12m$. In the absence of privileged information, we detect vehicles in BEV space by finding connected components in the BEV occupancy grid. Then, to enlarge vehicles, we continuously dilate the connected component until we surpass an area threshold. We experimentally set this area threshold to $8m^2$, corresponding to 200 pixels in our 5 pixels per meter setting. To randomly color objects, we similarly assign a random color per connected component. To maintain color per vehicle across time steps, we compare the current BEV to the previous time step and reuse the color of the previous connected component if the previous connected component covers over 50% of the area of the current connected component.

We provide the hyper-parameters for different versions of the model in Table 8. Due to computational requirements increasing with the number of slots, we proposed a lighter version of the base SAVi model we start with. In the light version, we reduced the number of parameters in the decoder and the encoder MLP. Using the lighter version, we train SAVi with up to 30 slots in our experiments. We highlight the differences between the versions in bold in Table 8.

Quantization: We quantize the traffic light status, speed, waypoints, and target points using k-means to assign them to the cluster index. For waypoints

Attribute	#Clusters (k_{attr})
Speed	14
Light Status	2
Waypoints	48 (24 per dim.)
Target Points	32 (16 per dim.)

Table 5: Number of Clusters used in K-means. We use separate clusters for each dimension in the case of multi-dimensional attributes. Although light status is binary, we use k-means with 2 clusters for consistency across attributes.

Hyper-param	Value
Hidden Dim.	768
MLP Dim.	3072
Attention Heads	12
Number of Layers	6
Weight Initialization	distilgpt2 [44]

Table 6: Hyper-parameters of the GPT-2 Backbone. We initialize the weights using the publicly available checkpoint of distilgpt2 [44].

and target points, which are both 2-dimensional, we apply k-means clustering on every dimension separately and quantize them into two indices in a similar manner. We list the number of clusters for each attribute in Table 5.

GPT Backbone: We use a slightly modified version of GPT-2 [41] as the backbone. We use the Transformers library [50] and modify the embedding layer and the attention layers to implement block attention. We list the hyper-parameters for the GPT-2 model used in Table 6.

2 Training Details

Hyper-param	Value
Epochs	100
Warmup Epochs	5
Learning Rate	5e-5
Optimizer	AdamW
Weight Decay	1e-4
Effective Batch Size	512
Grad. Norm Clip	1.0

Table 7: Training Hyper-parameters of CarFormer.

	SAVi-30-light	**SAVi-7-light**	**SAVi-7-base**
Effective Batch Size	256	256	256
Training Steps	80000	80000	80000
Optimizer	Adam	Adam	Adam
Learning Rate	1e-4	1e-4	1e-4
Gradient Clip	5e-2	5e-2	5e-2
Warmup steps	4000	4000	4000
Context Length	2 frames	2 frames	2 frames
Data FPS	2 fps	2 fps	2 fps
Num. Slots	**30**	**7**	**7**
Slot Size	128	128	128
Slot MLP size	256	256	256
Slot Attn. Iterations	2	2	2
Enc. Filter Sizes	(64, 64, 64, 64)	(64, 64, 64, 64)	(64, 64, 64, 64)
Enc. Kernel Sizes	(5, 5, 5, 5)	(5, 5, 5, 5)	(5, 5, 5, 5)
Enc. Strides	**(2, 1, 1, 1)**	**(2, 1, 1, 1)**	**(1, 1, 1, 1)**
Enc. MLP dim.	**128**	**128**	**256**
Dec. Init. Resolution	(128, 24, 24)	(128, 24, 24)	(128, 24, 24)
Dec. Filter Sizes	**(64, 32, 16, 8)**	**(64, 32, 16, 8)**	**(64, 64, 64, 64)**
Dec. Kernel Sizes	(5, 5, 5, 5)	(5, 5, 5, 5)	(5, 5, 5, 5)
Dec. Strides	(2, 2, 2, 1)	(2, 2, 2, 1)	(2, 2, 2, 1)
Pred. Type	Transformer	Transformer	Transformer
Pred. Layers	2	2	2
Pred. MLP Size	512	512	512
Pred. Attn. Heads	4	4	4

Table 8: Hyper-parameters of SAVi. We refer to the version of the encoder with a light decoder as SAVi-30-light and SAVI-7-light. We refer to the original SAVi model as SAVi-7-base. We show the hyper-parameters that are different across models in bold.

We train our model for 100 epochs on the full dataset and pick the checkpoint with the best validation loss to evaluate. We use a setup of either 4× A6000s or 4× A100s to train our models. We keep the effective batch size as 512 across runs. Moreover, we weigh the two loss terms by multiplying the forecasting loss by 40 as shown in (7). We list the other hyper-parameters in Table 7.

$$\mathcal{L} = 40\ \mathcal{L}_{forecast} + \mathcal{L}_{wp} \tag{7}$$

3 Additional Quantitative Results

Ablation of Light SAVi: We modify the architecture of SAVi to be able to increase the number of slots while keeping slot extraction feasible. Specifically, we propose a lighter decoder by decreasing the number of parameters. To measure the effect of these changes quantitatively, we train our model with the light decoder while keeping the number of slots constant at 7. Note that, training the

slot extraction model is prohibitively slow with the original decoder when we increase the number of slots to 30.

Method	#Slots	LD	#Steps	ARI↑	mIoU↑
SAVi	7	✗	-	0.905	0.856
	7	✓	-	0.911	0.860
	30	✓	-	0.924	0.874
CarFormer	7	✗	1	0.742	0.644
	7	✓	1	0.704	0.602
	30	✓	1	0.795	0.702
	7	✗	4	0.513	0.436
	7	✓	4	0.462	0.385
	30	✓	4	0.540	0.454
Input-Copy	-	-	1	0.641	0.561
	-	-	4	0.412	0.375

Table 9: Forecasting Results with Light Decoder (LD). We show the results for SAVi reconstruction, which serve as the upper bound as the model has access to ground truth. We compare six versions of our model, with each of the three SAVi versions (7-base, 7-light, 30-light), and trained to predict 1 or 4 timesteps into the future. Finally, we show the results of predicting the current BEV to forecast the future as a baseline.

We evaluate the forecasting performance of the model in Table 9. Although the performance degrades slightly with the light decoder in the case of 7 slots, increasing the number of slots to 30 improves performance noticeably. Increasing the number of slots is not only crucial to improve the performance in terms of slot extraction, it also proves to be crucial in terms of driving performance as we show in the main paper (Table 3). To further evaluate its effect on driving performance, we compare CarFormer using the original SAVi checkpoint with 7 slots (full decoder) as well as the light decoder version with 7, 14, and 30 slots in Table 10. The results show that increasing the number of slots improves the driving performance and the best driving performance is achieved with 30 slots which was only made possible with the light decoder.

The Effect of Enlarging Slots on Forecasting: Slot extraction is particularly difficult for small objects due to the nature of the reconstruction loss being dominated by larger objects. As a simple fix, we enlarge small vehicles in the BEV input before slot extraction. In particular, we enlarge any vehicle that is smaller than 4.9 m in length or 2.12 m in width to a length of 4.9 m and a width of 2.12 m, respectively. We set the dimensions according to Lincoln MKZ, the default ego vehicle on CARLA.

We evaluated the effect of enlarging slots on the driving performance in the main paper (Table 3). Here, we investigate its effect on forecasting. We compare two models, one with the base SAVi with 7 slots (SAVi-7-base), and our version

LD	#Slots	DS↑	IS↑	RC↑
×	7	60.82±0.87	0.73±0.02	79.44±1.62
✓	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 10: Driving Performance with Light Decoder (LD). We test three different SAVi versions. Light refers to the lighter version of SAVi we propose, and the base version otherwise. #Slots refers to the number of slots used in the model. We report mean±std of 3 different runs on the Longest6.

Method	#Slots	Enlarge	#Steps	ARI↑	mIoU↑
SAVi	7	×	-	0.900	0.846
		✓	-	0.905	0.856
CarFormer	7	×	1	0.712	0.611
		✓	1	0.742	0.644
		×	4	0.487	0.412
		✓	4	0.513	0.436
SAVi	30	×	-	0.910	0.850
		✓	-	0.924	0.873
CarFormer	30	×	1	0.765	0.664
		✓	1	0.795	0.703
		×	4	0.508	0.422
		✓	4	0.540	0.454

Table 11: Forecasting Results by Enlarging Slots. We compare the forecasting performance of the base SAVi model for the 7-slot case (SAVi-7-base) and the light version for the 30-slot case (SAVi-30-light) for predicting 1 or 4 steps ahead.

of SAVi with the light decoder and 30 slots (SAVi-30-light) in Table 11. Enlarging small vehicles consistently improves forecasting performance in all settings and in terms of all metrics.

The Effect of Block Attention on Slot Forecasting: In Table 2 of the main paper, we show that block attention (BA) between slots improves driving performance, and hypothesize that this allows us to better model the relationship between objects. To further validate the role of BA in relating objects, we evaluate the forecasting performance with and without BA in Table 12. The model with BA results in consistent improvements in forecasting performance with both 14 and 30 slots, which shows its positive effect by allowing more interaction between objects in the autoregressive model compared to regular causal attention. We outline how we can retain the ability to autoregressively rollout future time steps in Section 5.

BA	#Slots	ARI↑	mIoU↑
×	14	0.511	0.432
✓	14	0.533 (+0.022)	0.449 (+0.017)
×	30	0.509	0.428
✓	30	0.540 (+0.031)	0.454 (+0.026)

Table 12: Forecasting Results with Block Attention (BA). We evaluate CarFormer with our optimized SAVi with either 14 or 30 slots as the slot extractor backbone. The model with BA results in consistent improvements in all forecasting metrics.

Input Feature							
TP	S	TL	O	R	DS↑	IS↑	RC↑
×	×	×	✓	✓	74.89±1.44	0.79±0.02	92.90±1.28
×	×	×	✓	×	75.06±2.10	0.80±0.01	90.99±1.26
×	✓	✓	✓	×	70.01±2.79	0.79±0.02	86.23±1.71
✓	✓	✓	✓	✓	72.34±2.53	0.78±0.02	91.21±1.07

Table 13: Block attention design ablation. We compare placing different inputs into the attention block. Since our design contains only one block, a tick mark means the feature is included in the block. Results are of 3 runs on Longest6. TP: Target Point, S: Speed, TL: Traffic Light, O: Objects (attributes/slot features), R: Desired route.

Ablating the Design of Block Attention: In our main model, we place the object level features, whether in the form of slots or as attributes, in the same block as the desired route $\mathbf{r}_t^1, \mathbf{r}_t^2$. We ablate different possible options for the block in Table 13. Although removing the desired route from the block leads to slight performance gains in DS, we include the desired route in the block due to a lower standard deviation in driving score.

Ablation of Target Point Input: In our approach, we feed the next target point in the desired route as an input to the backbone. With the desired route being also given as input, this could possibly be redundant. We compare the results with and without the target point in Table 14. The inclusion of the next target point improves the DS, IS, and the RC of the agent.

TP	DS↑	IS↑	RC↑
✓	74.89±1.44	0.79±0.02	92.90±1.28
×	72.34±0.35	0.77±0.00	90.70±0.66

Table 14: Difference with versus without target point as input We compare CarFormer with and without using target points as input. Results are of 3 runs on Longest6.

Increasing the Input Context Size: In all the experiments in the main paper, the model only sees the information from the current time step. Although BEV from previous time steps is used within SAVi, we only use the slot features of the current time step. To explore the effectiveness of extending our model to multiple time steps, we train the model using a context length of up to 4 and evaluate the forecasting performance in Table 15. We observe consistent gains in both forecasting metrics as we increase the context length, showing the model is able to utilize information from previous timesteps to guide its future prediction.

Context Length					
1		**2**		**4**	
ARI	**mIoU**	**ARI**	**mIoU**	**ARI**	**mIoU**
0.795	0.702	0.816	0.728	0.824	0.740

Table 15: Effect of input context length on forecasting: Increasing the context length consistently improves forecasting performance in all metrics, showing that the model is able to use the information from previous timesteps to better predict the future.

Effect of Forecasting Steps: To further investigate the effect of forecasting, we experiment with the number of steps into the future that we forecast or f in (4). As can be seen in Table 16, increasing the number of forecasting steps results in improvements in both infraction score and route completion, and consequently a higher driving score. As a result, unless otherwise specified, all our models have been trained to predict timestep $t + 4$.

Effect of Forecasting Weight: We evaluate the effect of varying the weight of the forecasting term, α in the loss function (6) on driving performance in Fig. 3. While initially increasing the importance of forecasting improves the driving performance, the performance peaks around 40 and drops for larger values of α. This could be because the training signal from forecasting eventually dominates the parameter updates to the model, reducing the relative importance of the actual driving task.

Efficiency: In Table 17, we list the inference time in milliseconds (ms) and the number of parameters in millions (M), trainable parameters in parenthesis, for different versions of our model compared to PlanT. We measure the inference time for all models on a machine with i9-10900K CPU and RTX 3090 GPU. For our models, we report the results with object attributes similar to PlanT, with VQ-VAE, and with 7, 14, and 30 slots. All our models can run in real-time, with over 60 FPS. We also report the results without SAVi to show that around half of the time is spent on slot extraction. The inference time can be significantly improved with more efficient slot extraction techniques.

#Steps	DS↑	IS↑	RC↑
1	71.40±0.74	0.78±0.01	89.25±0.99
4	74.89±1.44	0.79±0.02	92.90±1.28

Table 16: Varying Forecasting Time steps. We compare varying the number of timesteps into the future that we forecast.

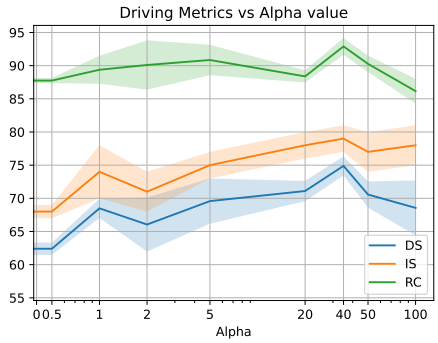

Fig. 3: Ablation of Forecasting Weight. We visualize the effect of varying the hyper-parameter α, the weight of forecasting in the loss function, on driving performance. While initially increasing the contribution of forecasting improves the driving performance, it peaks around 40. We use CarFormer with SAVi-30-light as the encoder backbone for all experiments.

4 Additional Qualitative Results

4.1 Effect of Increasing the Number of Slots

We visually compare the reconstructions of the different versions of SAVi to better understand the effect of increasing the number of slots. As can be seen in Fig. 4, the SAVi model we use with 7 slots cannot capture all the vehicles in the scene. Some vehicles are either missing completely as in b, e, or replaced with blurry regions as in a, f. The SAVi model with 14 slots faces the same issue, albeit to a lesser extent. For example, there is a missing vehicle in the top right of b, as well as multiple missing vehicles in e and f, including one very close to the ego vehicle in e, which could result in a collision.

When increasing the number of slots to 30, the model can capture all vehicles to some extent. One side effect of increasing the number of slots to 30 is that multiple slots can bind to a single vehicle, which can be seen in the reconstructions. For example, the vehicles in the bottom of c appear to be reconstructed as multiple blobs. However, the model can capture all vehicles in the scene, which empirically improved driving performance and infraction scores.

Model	Time (ms)	#Params
PlanT	6.41	42M
Obj. Attributes	7.31	53M
VQ-VAE	8.51	60M (43M)
7 Slots	13.87	54M (44M)
14 Slots	13.94	54M (44M)
30 Slots	14.02	54M (44M)
7 Slots w/o SAVi	7.46	54M (44M)
14 Slots w/o SAVi	7.37	54M (44M)
30 Slots w/o SAVi	7.44	54M (44M)

Table 17: Inference Time and Number of Parameters. We show inference time (in ms) and the number of parameters (in M) for different versions of our model compared to PlanT Medium [12]. When using slots as well as a VQ-VAE, we use a frozen backbone. In addition to the number of parameters, we show the number of trainable parameters in parenthesis.

4.2 Additional Slot Forecasting Visualizations

We provide additional samples of the model's future predictions in Fig. 5. In addition to future predictions using our main CarFormer model with SAVi-14-light, we also provide future predictions and SAVi reconstructions from our model with SAVi-7-base.

In b and i, the vehicle in the center of the scene is at an intersection. However, in the current timestep, the vehicle has yet to initiate the process of turning. Given only this information, the vehicle could end up taking either a right turn or a left turn. As a result, we see both models predicting a slight right turn while keeping the vehicle orientation the same. Due to the architecture of our model and the way future slot representations are predicted, we are not able to accurately capture multi-modality. As a result, the models predict the mean of two modes. This can also be seen with the vehicle on the left side in c and j making a right turn at the intersection.

The inability of the SAVi-7-base model to capture vehicles in scenes with many vehicles is highlighted in e and g. Due to the insufficient number of slots, the model outputs inaccurate predictions with blurry blobs. On the other hand, the SAVi-14-light model, with twice the number of slots, does not face the same problem as can be seen in l and m. The model accurately captures individual vehicles, and subsequently, predicts accurate future states.

5 Autoregressive Rollout with Block Attention

Using block attention breaks the causality within these blocks, which is typically required for generating rollouts autoregressively and reliably. If we limit block attention to the training only, we cause a distribution shift during test time

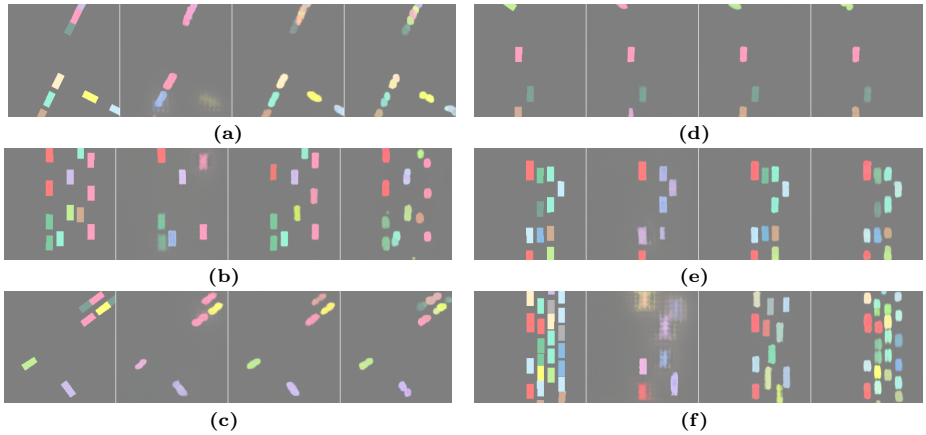

Fig. 4: Comparison of Slot Extraction by Varying the Number of Slots. Within each sub-figure, the columns correspond to ground truth BEV followed by SAVi-light reconstructions with 7, 14, and 30 slots, respectively.

which is not ideal. As a result, we briefly outline here how we can autoregressively generate rollouts, which is necessary for certain RL approaches such as Trajectory Transformer [24]. Note that to do this, we need to add a loss term to the loss function for next-word prediction, which is trivial as all inputs other than the scene representation are discretized.

In CarFormer, we use block attention exclusively in the scene representation part of the input. As this representation comes from another perception or feature extraction module, we can treat it as a unit. As a result, we assume we always begin with at least one set of features before we start generating rollouts. For the case of slots, at test time, we can start with the initial input trajectory as:

$$\tau_{inp} = \{g_t^x, g_t^y, l_t, v_t, \mathbf{z}_t^1, \ldots, \mathbf{z}_t^K, \mathbf{r}_t^1, \mathbf{r}_t^2\} \tag{8}$$

We would like to rollout future trajectories given this initial context. In this specific trajectory, we use block attention within the scene representation $\{\mathbf{z}_t^1, \ldots, \mathbf{z}_t^K, \mathbf{r}_t^1, \mathbf{r}_t^2\}$.

To begin, we can autoregressively predict the actions for the current timestep $\{q_t^1, \ldots, q_t^{2W}\}$ as well as the goal and initial part of the state of the next timestep $\{g_{t+1}^x, g_{t+1}^y, l_{t+1}, v_{t+1}\}$. This is a regular autoregressive next token prediction and does not require any modification. The only difference is that part of the input context contains block attention, which is similar to a variant introduced in [31] with *non-causal* attention within part of the prompt, typically the start of the prompt.

Next, to generate the slot features for time $t+1$, remember that we train our backbone with a forecasting objective for the scene-level representation at time $t+f$. As a result, we can do this if we set f to 1 when training the CarFormer

model. Given the forecasting head, we can get the next scene-level representation $\{\mathbf{z}_{t+1}^1, \ldots, \mathbf{z}_{t+1}^K\}$. Moreover, we also need to predict the route representations for the next time step, which can be done in the same way we currently predict slots to get $\{\mathbf{r}_{t+1}^1, \mathbf{r}_{t+1}^2\}$. Although we currently limit the forecasting loss to only slot features, nothing is limiting us from also predicting these next route features by modifying the forecasting loss. Finally, as we know the entire representation for the next scene representation at time $t+1$, we can append it to the trajectory and continue autoregressively predicting the next time steps as required.

One downside of predicting the next time step in language modeling entirely in one shot as a block is that the individual predictions are not conditioned on each other, especially because the outputs in language are typically discrete tokens that are either greedily chosen or sampled according to the output probabilities from the language modeling head. However, since object-level features are continuous in our case, this is likely less of a problem as there is no sampling step and the features interact with each other up to the attention layer in the last transformer layer. For the case of VQ-VAE and other discretized scene representations, we refer the reader to Non-Autoregressive Transformers (NATs), which are proposed in [19] for machine translation and other use cases.

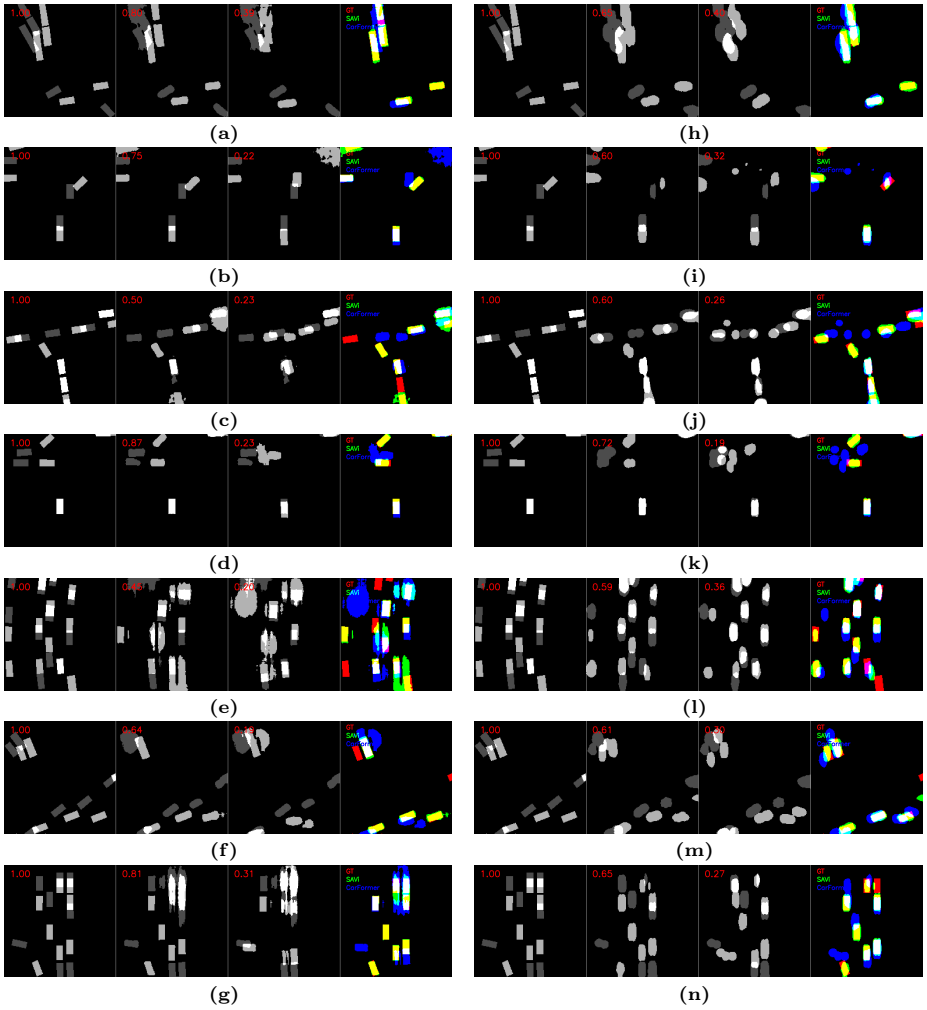

Fig. 5: Additional Visualization of Slot Forecasting Results for CarFormer with SAVi-7-base (left half) and CarFormer with SAVi-14-light (right half). Each sub-figure shows an example of input (dark grey)-output (light grey) objects in the first column, SAVi reconstructions in the second column, and 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 can be seen from combinations of R-G-B colors such as yellow (R+G) indicating misses and blue indicating false positives for our model (B).

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. Gu, J., Bradbury, J., Xiong, C., Li, V.O., Socher, R.: Non-autoregressive neural machine translation. In: ICLR (2018)
20. 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)
21. Harley, A.W., Fang, Z., Li, J., Ambrus, R., Fragkiadaki, K.: Simple-BEV: What really matters for multi-sensor bev perception? In: ICRA (2023)

22. 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)
23. 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
24. Janner, M., Li, Q., Levine, S.: Offline reinforcement learning as one big sequence modeling problem. In: NeurIPS (2021)
25. 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)
26. Karazija, L., Laina, I., Rupprecht, C.: Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. ARXIV **2111.10265** (2021)
27. 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)
28. Li, F., Kim, T., Humayun, A., Tsai, D., Rehg, J.M.: Video segmentation by tracking many figure-ground segments. In: CVPR (2013)
29. 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)
30. 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)
31. 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)
32. 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)
33. Micheli, V., Alonso, E., Fleuret, F.: Transformers are sample-efficient world models. In: ICLR (2023)
34. Mousavian, A., Fiser, M., Davidson, J., Kosecka, J., Toshev, A.: Visual representations for semantic target driven navigation. In: ICRA (2019)
35. Müller, M., Dosovitskiy, A., Ghanem, B., Koltun, V.: Driving policy transfer via modularity and abstraction. In: CORL (2018)
36. 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)
37. Ochs, P., Malik, J., Brox, T.: Segmentation of moving objects by long term video analysis. IEEE TPAMI (2013)
38. van den Oord, A., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: NeurIPS (2017)
39. 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)
40. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: ECCV (2020)
41. 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)
42. Ren, X., Wang, X.: Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In: CVPR (2022)

43. 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)
44. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In: NeurIPS EMC2 Workshop (2019)
45. Sauer, A., Savinov, N., Geiger, A.: Conditional affordance learning for driving in urban environments. In: CORL (2018)
46. 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)
47. 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)
48. Shafiullah, N.M.M., Cui, Z.J., Altanzaya, A., Pinto, L.: Behavior transformers: Cloning k modes with one stone. In: NeurIPS (2022)
49. Wang, D., Devin, C., Cai, Q.Z., Krähenbühl, P., Darrell, T.: Monocular plan view networks for autonomous driving. In: IROS (2019)
50. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-Art Natural Language Processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (2020)
51. Wu, Z., Dvornik, N., Greff, K., Kipf, T., Garg, A.: SlotFormer: unsupervised visual dynamics simulation with object-centric models. In: ICLR (2023)
52. 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)
53. 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`
54. Yan, W., Zhang, Y., Abbeel, P., Srinivas, A.: VideoGPT: video generation using VQ-VAE and transformers. ARXIV **2104.10157** (2021)
55. Yang, L., Fan, Y., Xu, N.: Video instance segmentation. In: CVPR (2019)
56. 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)
57. Zheng, Q., Zhang, A., Grover, A.: Online decision transformer. In: ICML (2022)
58. Zhou, B., Krähenbühl, P., Koltun, V.: Does computer vision matter for action? Science Robotics **4** (2019)