Deep Object Centric Policies for Autonomous Driving

Dequan Wang[1], Coline Devin[1], Qi-Zhi Cai[2*], Fisher Yu[1], Trevor Darrell[1]

***Abstract*— While learning visuomotor skills in an end-to-end manner is appealing, deep neural networks are often uninterpretable and fail in surprising ways. For robotics tasks, such as autonomous driving, models that explicitly represent objects may be more robust to new scenes and provide intuitive visualizations. We describe a taxonomy of "object-centric" models which leverage both object instances and end-to-end learning. In the Grand Theft Auto V simulator, we show that object centric models outperform object-agnostic methods in scenes with other vehicles and pedestrians, even with an imperfect detector. We also demonstrate that our architectures perform well on real world environments by evaluating on the Berkeley DeepDrive Video dataset.**

I. INTRODUCTION

End-to-end approaches to visuomotor learning are appealing in their ability to discover which features of an observed environment are most relevant for a task, and to be able to exploit large amounts of training data to discover both a policy and a co-dependent visual representation. Yet, the key benefit of such approaches—that they learn from task experience—is also their Achilles heel when it comes to many real-world settings, where behavioral training data is not unlimited and correct perception of "long-tail" visual phenomena can be critical for robust performance.

Learning all visual parameters of a visuomotor policy from task reward (or demonstration cloning) places an undue burden on task-level supervision or reward. In autonomous driving scenarios, for example, an agent should ideally be able to perceive objects and vehicles with a wide range of appearance, even those that are not well represented in a behavioral training set. Indeed, for many visuomotor tasks, there exist datasets with supervision for perception tasks, such as detection or segmentation, that do not provide supervision for behaviour learning. Learning the entire range of vehicle appearance from steering supervision alone, while optimal in the limit of infinite training data, clearly misses the mark in many practical settings.

Classic approaches to robotic perception have employed separate object detectors to provide a fixed state representation to a rule-based policy. Multistage methods, such as those which first segment a scene, can avoid some aspects of the domain transfer problem [1], but do not encode discrete objects, and thus are limited to holistic reasoning. End-to-end learning with pixel-wise attention can localize specific objects and provide interpretability, but throws away the existence of instances.

[1]EECS Department, University of California, Berkeley
[2]CS Department, Nanjing University
*Work done while at UC Berkeley

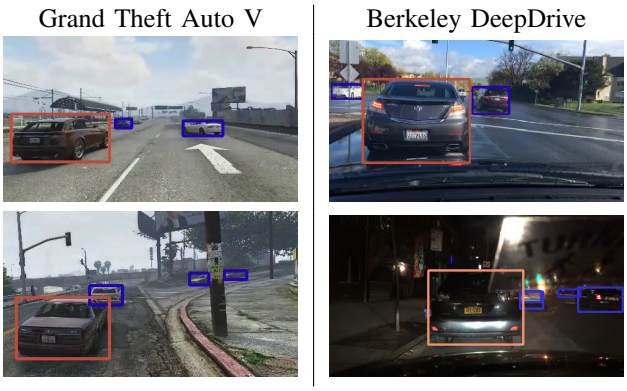


Fig. 1. Our method uses discrete objects as part of the policy model for driving in traffic. The learned selector identifies the objects most relevant to the policy, which is often the nearest car.

We propose an object-centric perception approach to deep control problems, and focus our experimentation on autonomous driving tasks. Existing end-to-end models are holistic in nature; our approach augments policy learning with explicit representations that provide object level attention.

In this work we consider a taxonomy of representations that consider different levels of objects-centric representations, such as discreteness and sparsity. We define a family of approaches to object-centric models, and provide a comparative evaluation of the benefit of incorporating object knowledge either at a pixel or box level, with either sparse or dense coverage, and with either pooled or concatenated features.

We evaluate these aspects in a challenging simulated driving environment with many cars and pedestrians, as well as on real dash-cam data, as shown in Figure 1. We show that using a sparse and discrete object-centric representation with a learned per-object attention outperforms previous methods in on-policy evaluations and provides interpretability about which objects were determined most relevant to the policy.

II. RELATED WORK

Approaches to robot skill learning face bias/variance trade-offs, including in the definition of a policy model. One extreme of this trade-off is to make no assumptions about the structure of the observations, such as end-to-end behavior cloning from raw sensory data [3], [4], [5]. At the opposite end, one can design a policy structure that is very specific to a particular task, *e.g.* for driving by calculating margins between cars, encoding lane following, and tracking pedestrians [6]. These modular pipelines with rule-based system dominate autonomous driving industry [7], [8], [9].

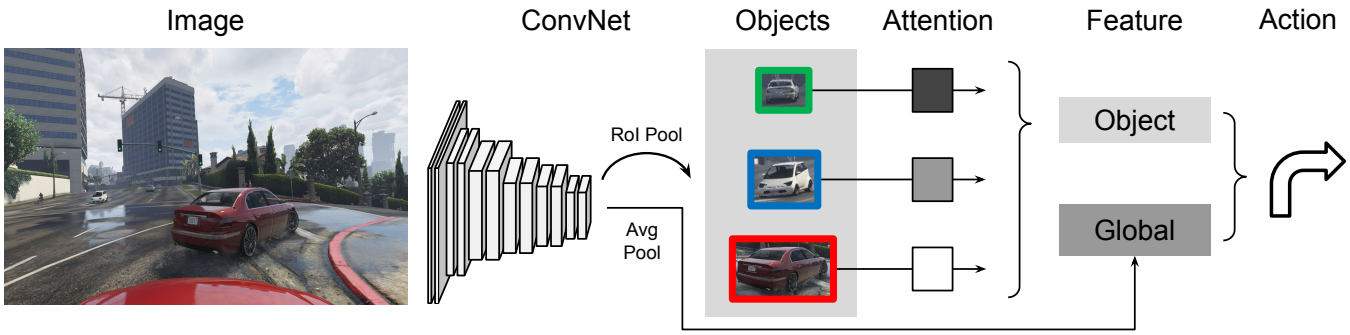


Fig. 2. The overview of object-centric architecture. The image is first passed through a 34-layer DLA convolutional network [2], which outputs RoI pooled features for each object along with globally pooled features for the whole image. Then object-level attention layer calculates the task-oriented importance score for each RoI. The linear policy layer takes both global and object features and predicts action for next step.

Fig. 3. An illustration of the representation taxonomy we describe in Section III-B. (a) shows a global image representation that does not leverage objects. (b) is a continuous (pixel-level) attention that selects salient parts of the image. (c) is a dense and discrete object representation that selects all objects in the scene. (d) is a discrete but sparse object presentation that only selects the objects important for the task. (e) is a sparse representation that treats each object individually by concatenating instead of averaging the object features.

The first attempt at training an end-to-end driving policy from raw inputs traces back to 1980s with ALVINN [10]. Muller *et al*. revisited this idea to help off-road mobile robots with obstacle avoidance system [11]. Recently, Bojarski *et al*. demonstrate the appeal of foregoing structure by training a more advanced convolutional network to imitate demonstrated driving [3], [4]. Xu *et al*. advocate learning a driving policy from a uncalibrated crowd-sourced video dataset [5] and show their model can predict the true actions taken by the drivers from RGB inputs. Codevilla *et al*. [12] leverage the idea of conditional imitation learning on high-level command input in order to resolve the ambiguity in action space. These end-to-end models, which automatically discover and construct the mapping from sensory input to control output, reduce the burden of hand-crafting rules and features. However, these approaches have not yet been shown to work in complex environments, such as intersections with other drivers and pedestrians.

We address how to best represent images for robotics tasks such as driving. Muller *et al*. train a policy model from the semantic segmentation of images, which increases generalization from synthetic to real-world [1]. Chen *et al*. provide an additional intermediate stage for end-to-end learning, which learns the policy on the top of some ConvNet-based measurements, such as affordance of road/traffic state for driving [13]. Sauer *et al*. combine the advantages of conditional learning and affordance [14]. The policy module is built on a set of low-dimensional affordance measurements, with the given navigation commands. We argue for an object-centric approach which allows objects to be handled explicitly by the model. Prior work has encoded objects as bounding box positions [15] for manipulation tasks, but does not use end-to-end training and discards all information about the objects except for their pixel positions. We expand upon this work and evaluate a taxonomy of "object-centric" neural network models on the driving task.

III. OBJECT-CENTRIC POLICIES

We describe an generic architecture that takes in RGB images and outputs actions. Our model expresses a series of choices that add different levels of object-centricity to the model. Our goal is to identify which aspects are important for visuomotor tasks such as autonomous driving.

A. Generic Architecture

The generic form of our model takes in an RGB image and outputs two sets of features: global image contextual features and an object-centric representation. The global contextual features are produced by a convolutional network over the whole image, followed by a global average pooling operation. The object-centric representation is constructed as described below to produce a fixed-length object-centric representation. The global features are concatenated with the object representation, and passed to a fully connected policy network which outputs a discretized action. For on-policy evaluation, a hard-coded PID controller converts the action to low-level throttle, steer, and brake commands.

Algorithm 1 Discrete Object Centric Architecture: the model may be sparse or dense, and use concatenation or summation. For a sparse model, k is the number of objects to keep.

I is received from the sensors
G := GlobalFeatures(I)
O, N := Detector(I) // N is the number of objects detected
for $0 \leq i < N$ **do**
    f_i := RoI(o_i) // Object features
    w_i := Selector(f_i, G) // Object score
end for
$\bar{w}_0, ...\bar{w}_N$ = Softmax($w_0, ..., w_N$)
$f_i := \bar{w}_i * f_i$ for all i
if sparsity **then**
    Sort objects by $\bar{w}$ and keep only the top k
end if
if concatenation **then**
    return concatenate(remaining f_i, sorted by w_i)
else if summation **then**
    return sum(remaining f_i)
end if

B. Objectness Taxonomy

What does it mean for a end-to-end model to be "object-centric"? In this section, we define a taxonomy of structures that leverage different aspects of "objectness". By defining this taxonomy and placing previous work within it, we evaluate which aspects bring the greatest gains in performance specifically for driving scenarios. The aspects discussed are *countability*, *selection*, and *aggregation*. Figure 3 visualizes the levels.

1) Countability: Discrete vs Continuous: An example of a continuous object-centric representation is a pixel-level attention map over an image, as used in [16]. In contrast, a discrete representation could be a bounding box or instance mask. The potential benefit of keeping a discrete object structure is that a model may need to reason explicitly over instances (such as cars navigating an intersection) rather than reasoning over the global vehicle "stuff". Our implementation of discrete objects applies pre-trained FPN detector [17] to output bounding boxes for vehicles and pedestrians. We utilize RoI-pooling layer [18] to extract regional feature for each box. The boxes and their respective features are treated as a set of objects. In the discrete setting, we define O as the list of objects returned by the detector, and $f(o_i)$ as the RoI features of the i-th object. We define G as the global features from the whole image.

2) Selection: Sparse vs Dense: Should the policy model reason over all objects at once (dense), or should it first select a fixed number (sparse) of salient objects and consider only those? The former allows more flexibility, but e.g., may distract the policy with cars that are very far away or separated from the agent by a median. To obtain a relevance score for each object, we train a task-specific selector jointly with the policy. The selector is a network that takes in the RoI features of each object concatenated with the global image features and outputs a scalar score, indicating the relevance of the object. The scores w are evaluated with a softmax to produce a weight between 0 and 1 for each object. In the sparse model, only the top k scoring objects are used in the policy.

3) Aggregation: Sum vs Concatenate: If using discrete objects, a decision needs to be taken about how to combine the objects into a single representation. One possible approach is the weight and sum the features of the objects, while another approach is to concatenate the features. The former is agnostic to the number of objects and is order invariant, while the latter may allow for more nuanced computation about multi-object decisions. Our implementation of the concatenation approach is to sort the objects by their selector weights and concatenate the features $\bar{w}_i * f_i$ in order from largest w_i to smallest.

IV. EXPERIMENTS

We evaluate our object-centric models on both a simulated environment and a real-world dataset. Specifically, we use the Grand Theft Auto V simulation [19] and the Berkeley DeepDrive Video dataset [5] for online and offline evaluation, respectively. All models are trained on a behavioral cloning objective.

A. Evaluation Setup

1) Online Driving Simulation: For the simulation experiments, 1.6 million training frames were collected by using the in-game navigation system as the expert policy. Following a DAgger-like [20] augmented imitation learning pipeline, noise was added to the control command every 30 seconds to generate diverse behavior. The noisy control frames and the following ~ 7 frames were dropped during training to avoid replicating noisy behavior. The simulation was rendered at 12 frames per second. The training dataset was collected over 1000 random paths across 2km in the game. The in-game times ranged from 8:00 am to 7:00 pm with the default cloudy weather. In total, Each frame included control signals, such as speed, angle, throttle, steering, brake, as well as ground-truth bounding boxes around vehicles and pedestrians. During our training and testing procedure we used a camera in front of the car which keeps a fixed 60° horizontal field of view (FoV). The maximum speed of all vehicles was set to 20km/h.

When training a policy, the expert's continuous action was discretized into 9 actions: (*left, straight, right*) × (*fast, slow, stop*). At evaluation time, we used a PID controller to translate the discrete actions into continuous control signals per frame.

For testing, we deployed the model in 8 locations unseen during training, 2 highway and 6 urban intersections. Figure 7 demonstrates some example scene layouts in our simulation environment. For each location, we tested the model for 100 minutes: the agent was run for 10 independent roll-outs lasting 10 minutes each. If the vehicle crashed or got stuck during a rollout, the incident was recorded and the in-game AI intervened over for at least 15 seconds until

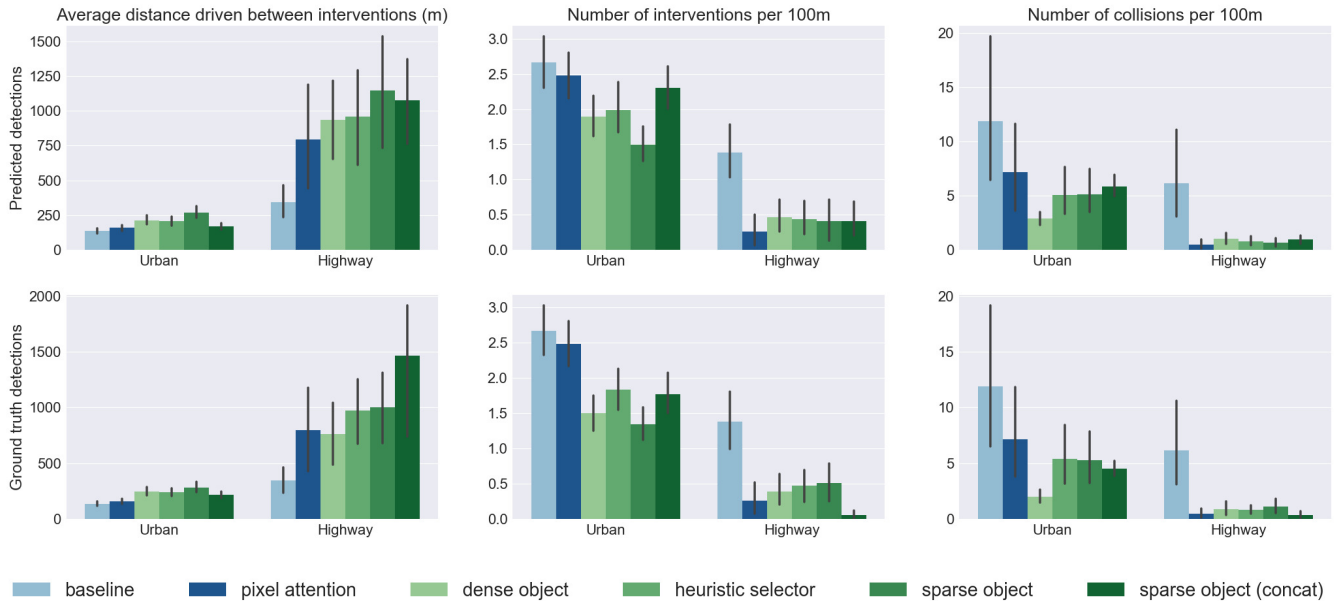


Fig. 4. Driving performance. From left to right: driving distance between interventions, number of interventions per 100m , number of collisions per 100m. The top row shows results using a learned detection model, while the bottom row uses ground-truth bounding box. The object centric models (green) overall perform better than the object agnostic models (blue), with the sparse models being the best. The highway environment is easier to drive than the urban environment. Comparing the heuristic selector with the learned selector used in the "sparse object" model, it is clear that learning a selector provides better results.

it recovered. An extreme accident which took more time to recover from would be penalized more in our metric as it would travel less far overall; the frames during the intervention were not counted towards the total.

The models were evaluated with several metrics. For each roll-out, we calculated the total distance travelled, the number of collisions, and the number of interventions by the in-game AI. To compare across roll-outs, we computed the distance driven between AI interventions, the number of collisions and interventions per 100m traveled.

2) Real-world Offline Dataset: We used 2.2 million training frames and 0.2 million testing frames from a large-scale crowd-sourcing dash-cam video dataset, with diverse driving behaviors. Each frame was accompanied by raw sensory data from GPS, IMU, gyroscope, magnetometer, as well as sensor-fused measurements like course and speed.

We follow the settings of continuous action driving model [5]. For each frame, the model was trained to predict the expert's future linear and angular speeds. The predictions were made at intervals of $1/3$ seconds during training.

For evaluation, we again follow the method in [5], which first discretized speed and angle into 30 bins each. We then mapped the joint distribution of speed and angle into $30 \times 30 = 900$ bins. We evaluated the 900-way classification model trained on this dataset by the perplexity of the model on withheld test data. Specifically, we calculated the value of softmax loss function as perplexity indicator, following the evaluation protocol of [5].

B. Implementation Details

The convolutional network was a 34-layer DLA model [2] pre-trained on ImageNet [21], with the open-source framework PyTorch [22]. We use a Detectron model [23] trained on MSCOCO [24] to generate bounding boxes for moving objects, specifically vehicles and pedestrians. We used the Adam optimizer [25] for 3 epochs with initial learning rate 0.001, weight decay 10^{-4}, and batch size 128. We do not use any data augmentation, which is different from [12], [14]. All sparse models use $k = 5$ to keep the top 5 objects and discard the rest.

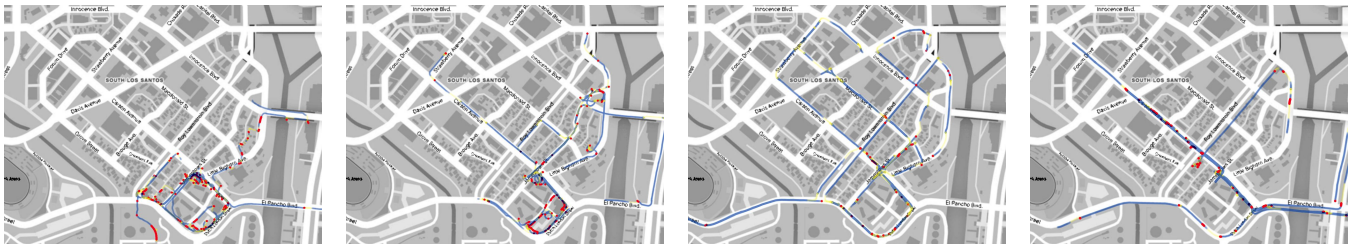
% data trained on	5%	10%	25%	50%	100%
baseline	2.52	2.40	2.29	1.94	**1.80**
pixel attention	2.70	2.33	2.15	1.96	1.84
dense object	2.34	2.24	2.07	2.06	2.01
heuristic selector	2.48	2.39	2.31	2.13	2.10
sparse object	**2.31**	**2.23**	2.19	2.07	2.10
sparse object concat	2.37	2.31	**2.04**	**1.93**	1.82

TABLE I

SPARSE TRAINING REAL WORLD EVALUATION. TO EVALUATE THE MODELS TRAINED ON REAL IMAGES, WE MEASURE THE PERPLEXITY OF THE MODELS ON WITHHELD TEST DATA AS AN OFF-POLICY EVALUATION. LOWER PERPLEXITY INDICATES THAT DATASET WAS MODELED MORE ACCURATELY.

C. Results

We evaluate several baselines, prior methods, and ablations. The *baseline* method is based on the the network by Xu et al. [5], which does not represent objects or use attention at inference time. The *pixel attention* method is the same as *baseline* but with an additional pixel-level attention mechanism, learned end-to-end with the task. This



(a) baseline (b) pixel attention (c) dense object (d) sparse object

Fig. 5. Sample trajectories from the evaluation. Yellow dots indicate to interventions while red dots indicate collisions (best viewed on screen). This example illustrates the reliability of the object centric models over the the baselines, with less collisions & interventions and more travelling distances.

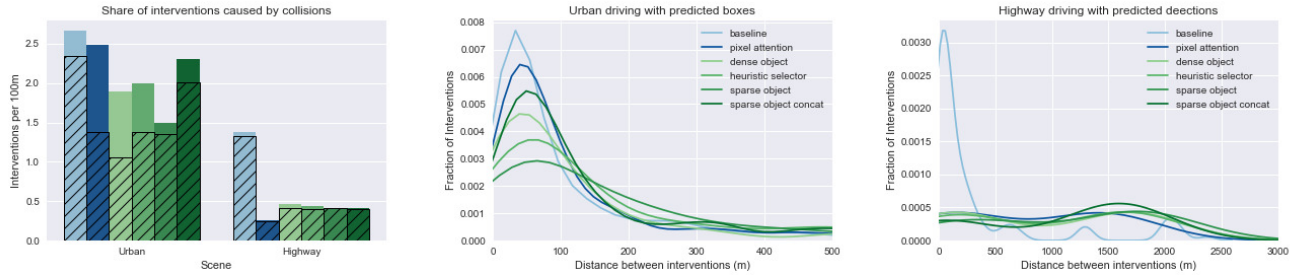


Fig. 6. Analysis of intervention frequency. On the left, the shaded region measures the proportion of interventions caused by collisions. In the highway environment, almost all interventions are cause by collisions, but in the more complex urban environment, the policy get stuck at an intersection as shown in the supplementary video. On the right, histograms shows how far each model drove between interventions and collisions. We see that in the urban environment, the object centric approaches drove farther in between interventions than the pixel attention or the baseline. In the highway environment, the pixel attention performs slightly better, probably because this environment does not require much navigation between cars and pedestrians.

is similar to [16]. Next, we evaluate several object-centric models drawn from our taxonomy. The results labeled *dense object* use a discrete and dense object representation with summation of the objects weighted by a learned selector. *Sparse object* is the same as *dense object*, but only looks at the top 5 objects in the scene, as scored by the learned selector. While the preceding models used the selector to weight object features before summing them, *sparse object concat* concatenates the features of the top 5 objects and passes the entire list to the fully connected policy. We also evaluate our selector by comparing to a heuristic selector: the size of the object's bounding box. The results using the heuristic selector in a sparse object model are labeled *heuristic selector*.

The results of the on-policy simulated driving are shown in Figure 4. We show several metrics: the number of collisions, the number of times the agent got stuck, and the distance driven between these. Each evaluation was repeated for two environments: urban (which has many intersections and cars/pedestrians) and highway (which is mostly driving straight). The object-centric methods consistently outperform the two object-agnostic method in the urban evaluation, while the highway environment shows good performance for all attentional models.

The comparable performance between the evaluation with ground truth boxes versus predicted boxes (from a detector trained on MSCOCO [24]) indicates that our method is robust to noisy detections. Figure 5 visualizes evaluation rollouts along a map with collisions and interventions drawn in. These maps show how the object centric models drive for longer without crashing or getting stuck, and how they end up farther from their start point than the baseline and pixel attention models. This is supported by the histograms of distance between interventions in Figure 6 which shows how the sparse models especially drive farther between interventions.

To identify the benefits of using a learned selector over boxes, we compared the *sparse object* model against a heuristic selector, which assigns importance to objects based on their size. The motivation for this heuristic is that larger objects are likely to be closer, and therefore more important for the policy. Figure 4 shows that the model with a learned selector performs equally or better than the heuristic for every metric. Although some other heuristic may work better, we conclude that learning the selector jointly with the policy is beneficial.

The final experiment in Table I is an off-policy evaluation on the real world dataset that measures the perplexity of the learned model with respect to test data. When trained on only a subset of the data (from 5% to 50%), the sparse object models performs best, with concatenation overtaking summation in the medium data regime. The concatenation model performs equally well to the baseline once all the data has been seen, indicating that the sparse model is advantageous for low data problems, and that the sparse concat model is ideal for medium to large data situations. The object prior that our models leverage allows them to learn quickly from little data without being distracted by irrelevant pixels. Figure 8 shows example scenes with our model's attention.

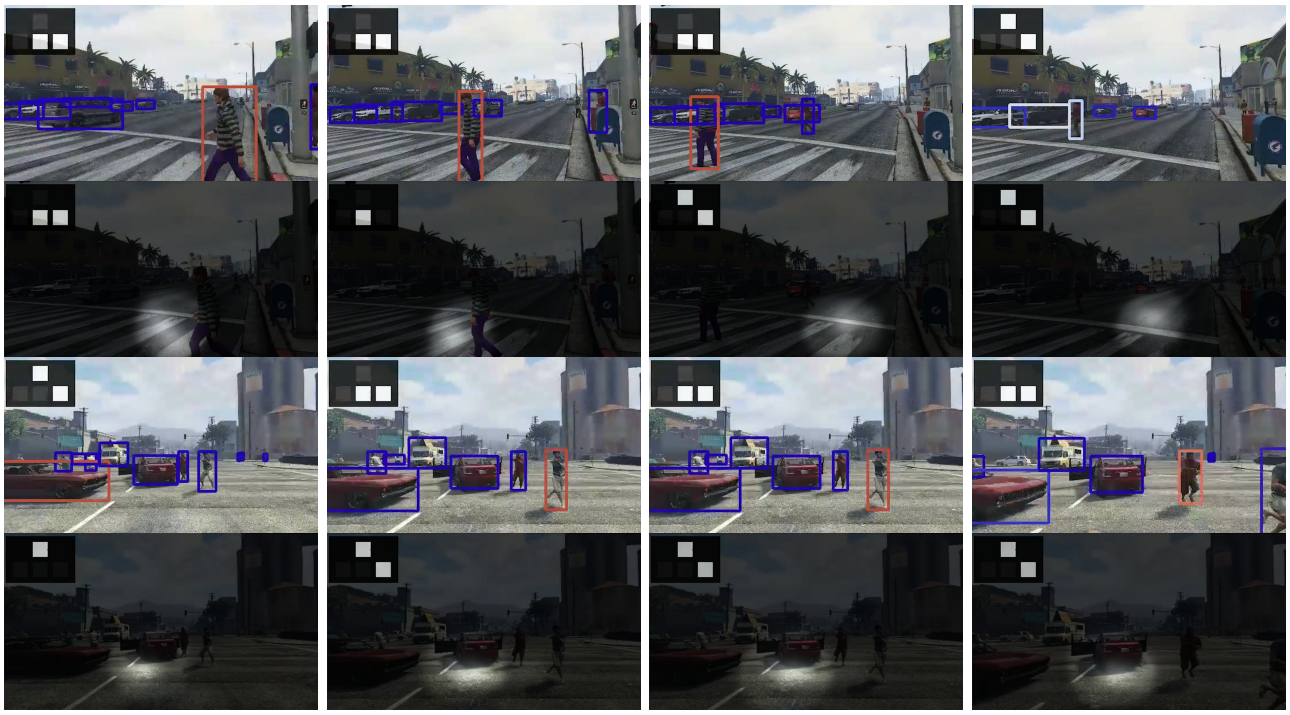


Fig. 7. Sample scenes from the Grand Theft Auto V simulation with our sparse model's learned object selector compared against a learned pixel-level attention. For rows 1 and 3, red indicates a high scoring object, and blue is low scoring (best viewed on screen). For rows 2 and 4, then pixel attention is shown by brightness of the pixels. The actions output by each model are shown by the white squares in the corners: accelerator is the top square, and the bottom squares are turn left, brake, and turn right, respectively. A single action may both turn and accelerate or brake. Rows 1 and 2 shows both models performing well, while rows 3 and 4 show the pixel attention model ignoring pedestrians and deciding to accelerate towards them. The object centric model is more conservative and attends strongly to the pedestrians, choosing to slow down instead of speeding up.

Fig. 8. Sample scenes from the Berkeley DeepDrive Video dataset with the sparse model's learned selector visualized. Red indicates a high scoring object, and blue is low scoring (best viewed on screen). Our method is robust to imperfect detections, such as overlapping bounding boxes, for both both day and night scenes.

V. CONCLUSION

We defined a taxonomy over object-centric models and showed in an on-policy evaluation that sparse object models outperformed object-agnostic models according to our metrics of distance driven and frequency of collisions and interventions. Our results show that highway driving is significantly easier than navigating intersections; the necessity of navigating city environments showcase the advantages of representing objects. Overall the results, discreteness and sparsity, along with a learned selection mechanism, seem to be the most important aspects of object-centric models.

For simplicity, this work only considered the presence of vehicles and pedestrians and did not evaluate the policies ability to follow the rules of the road. Using generic object detection rather than class specific detection would hopefully lead to paying attention to streetlight, signage, and other objects relevant to driving. These types of objects are crucial for following the rules of the road, and we expect that object-centric policies would provide even more gains in future settings. Promising avenues for future work also include leveraging the 3D nature of objects and their temporal coherence.

REFERENCES

[1] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *arXiv preprint arXiv:1804.09364*, 2018.

[2] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *CVPR*, 2018.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[5] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *CVPR*, 2017.

[6] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[7] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, 2006.

[8] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, 2008.

[9] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive-an autonomous journey on a historic route." *ITSM*, 2014.

[10] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NIPS*, 1989.

[11] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *NIPS*, 2006.

[12] F. Codevilla, M. Müller, A. Dosovitskiy, A. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *arXiv preprint arXiv:1710.02410*, 2017.

[13] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *ICCV*, 2015.

[14] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," *arXiv preprint arXiv:1806.06498*, 2018.

[15] C. Devin, P. Abbeel, T. Darrell, and S. Levine, "Deep object-centric representations for generalizable robot learning," in *ICRA*, 2017.

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

[17] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *CVPR*, 2017.

[18] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.

[19] P. Krähenbühl, "Free supervision from video games," in *CVPR*, 2018.

[20] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[23] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.