StopNet: Scalable Trajectory and Occupancy Prediction for Urban Autonomous Driving

Jinkyu Kim*[1], Reza Mahjourian*[2], Scott Ettinger[2], Mayank Bansal[2], Brandyn White[2], Ben Sapp[2], and Dragomir Anguelov[2]

***Abstract*— We introduce a motion forecasting (behavior prediction) method that meets the latency requirements for autonomous driving in dense urban environments without sacrificing accuracy. A whole-scene sparse input representation allows StopNet to scale to predicting trajectories for hundreds of road agents with reliable latency. In addition to predicting trajectories, our scene encoder lends itself to predicting whole-scene probabilistic occupancy grids, a complementary output representation suitable for busy urban environments. Occupancy grids allow the AV to reason collectively about the behavior of groups of agents without processing their individual trajectories. We demonstrate the effectiveness of our sparse input representation and our model in terms of computation and accuracy over three datasets. We further show that co-training consistent trajectory and occupancy predictions improves upon state-of-the-art performance under standard metrics.**

I. Introduction

An Autonomous Vehicles (AV) needs to continuously evaluate the space of all possible future motions from other road agents so that it can maintain a safe and effective motion plan for itself. This motion forecasting and re-planning task is one of the many processes that are continuously executed by the AV, so it is critical that it completes under expected latency requirements. On the other hand, operating in dense urban environments, the AV may encounter scenes with hundreds of dynamic agents within its field of view—consider driving next to a sports or music venue with lots of pedestrians. Autonomous driving in such environments requires a motion forecasting and planning system that is ① fast, ② scales well with the number of agents.

The existing motion forecasting methods do not meet the requirements discussed above. Models typically take upwards of 40-50ms for inference. This scalability issue is not addressed in public benchmarks [1], [2], [3], [4] and is often ignored in publications. Proposed methods often use *raster* (render-based) input representations [5], [6], [7], [8], which require costly CNNs for processing. Recently, methods have been proposed that use *sparse* point-based input representations [9], [10], [11], [12]. These methods offer improvements in accuracy and a reduction in the number of model parameters. However, with a focus on accuracy, these methods use *agent-centric* scene representations, which require re-encoding road points and agent points from the view point of each individual agent. The latency of these methods grows linearly with the number of inference agents, so they are not suitable for busy urban environments.

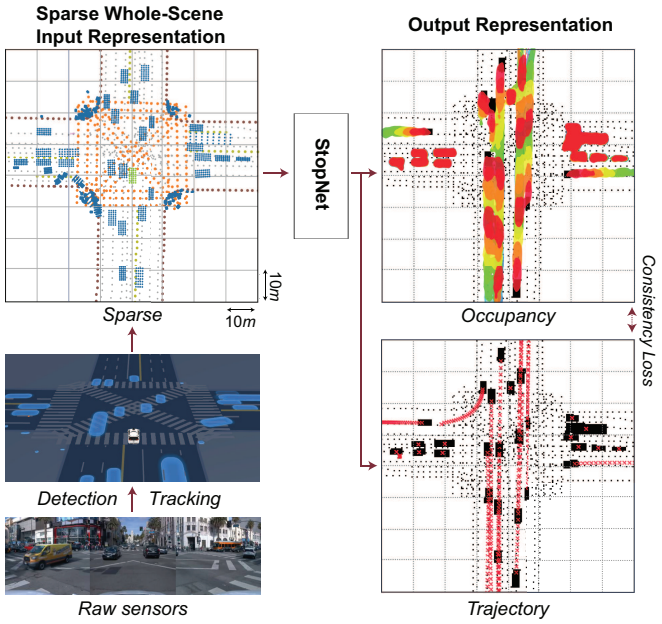

Fig. 1. StopNet uses a whole-scene sparse input representation, supporting a scalable motion forecasting model that unifies occupancy grids and trajectories.

This work introduces StopNet, a motion forecasting method focused on latency and scalability. We develop a novel *whole-scene* sparse input representation which can encode scene inputs pertaining to all agents at once. Drawing from the 3D object detection literature, we develop a PointPillars-inspired [13] scene encoder to concurrently process sparse points sampled from all agents, leading to a very fast trajectory prediction model whose latency is mostly invariant to the number of agents.

The predicted trajectories and uncertainties are often consumed as planning constraints by the AV, therefore the latency of the planning algorithm also increases in busy scenes. StopNet's whole-scene encoder also supports predicting probabilistic occupancy grids [14]—a dense output format capturing the probability that any given grid cell in the map is occupied by some agent part. This output representation allows the AV planner to reason about the occupancy of entire regions in busy scenes without a need for processing individual trajectories—thereby requiring almost constant computation. Another attractive property of occupancy grids is that they are robust to detection and tracking noise and flicker, since they allow the model to infer

*Equal contribution. Correspondence: rezama@waymo.com
[1]Department of Computer Science and Engineering, Korea University, Seoul 02841, South Korea. Work done while at Waymo.
[2]Waymo, Mountain View, CA 94043, USA.

occupancy independently of agent identity over time.

Via a co-training setup, StopNet is also the first method to unify trajectory sets and occupancy grids as the two archetypes of motion forecasting. We tie together these output representations with an intuitive consistency loss: the per-agent trajectory output distribution, when converted to an occupancy probability distribution, should agree with the overall occupancy distribution. Our experiments show that co-training in this manner leads to state-of-the-art trajectory prediction performance.

II. Related Work

Agent-Centric vs. Whole-Scene Modeling. While there are other alternatives, most prediction methods rely on a sequence of agent state observations often provided by a detection/tracking system [15]. Agent-centric models re-encode the world from the view point of every agent in the scene [9], [10], [16], [17], [6], [11], [18], [12], [19], [20]. This process requires transforming road state and the state of all other agents into an agent-centric frame. Therefore, these methods scale linearly with the number of agents, which poses a scalability issue in dense urban scenes with hundreds of pedestrians and vehicles. A popular alternative is whole-scene modeling [5], [8], [7], [21], [22], [23], where the bulk of the scene encoding is done in a shared coordinate system for all agents. Whole-scene modeling has the very attractive advantage that the processing time is invariant to the number of agents.

Dense vs. Sparse Input Representation. To our knowledge, whole-scene models have always used a bird's-eye view (BEV) raster input representation to encode road elements, agent state, and agent interactions. This approach allows including a variety of heterogeneous inputs into a common raster format, and enables the use of well-established powerful CNN models. However, there are several disadvantages. The model's field of view (FOV) and resolution are constrained by the computational budget, and the ability to model spatially-distant interactions is dependent on the receptive field of the network. Finally, while it is possible to render some state attributes, *e. g.*, vehicle extent, it is unclear how to rasterize some attributes, like uncertainty over agent orientation. On the other hand, with sparse inputs representations [11], [10], [9], [12] the model inputs consist of vectors of continuous state attributes encoding the agent motion history, relation to road elements, and relation to neighboring agents. This allows for arbitrary long-range interactions, and infinite resolution in continuous state attributes. However, sparse inputs have always been combined with agent-centric models, posing scalability issues. StopNet is the first method to address scalability by introducing a whole-scene sparse input representation and model.

Trajectory vs. Occupancy Output Representation. Representing future motion is traditionally done in two ways. The popular approach is a parametric distribution over a set of trajectories per agent [5], [9], [16], [17], [11], [18], [7], [21], [22], [23]. A common approach to capturing trajectory uncertainty is to predict multiple trajectories per agent as well as Gaussian position uncertainty for each trajectory waypoint, which in busy scenes, amounts to a large set of constraints to process in the planning algorithm. Moreover, the per-agent trajectories may be overlapping in space, and sampling from them independently may produce samples which violate physical occupancy constraints by placing agents on top of each other. An alternative output representation is to predict the collective occupancy likelihood as discretized space-time cells in a grid view of the world [6], [24], [8], [25], [26], [27]. While occupancy grid models have been mentioned in passing [8] and embedded in other tasks [27], in this work we study them in detail and develop metrics to evaluate them.

III. Method

A. Problem Definition

We assume that each agent at any time t can be represented by an oriented box as a tuple $(\mathbf{s}_t,\ \theta_t,\ w_t,\ l_t,\ v_t,\ a_t)$, where $\mathbf{s}_t = (x_t, y_t)$ denotes the agent's 2D center position, θ_t denotes the orientation, $(w_t,\ l_t)$ denote box extents, and $v_t,\ a_t$ denote 2D velocity and acceleration vectors. Given a sequence of state observations over a fixed number of input timesteps for all agents in the scene, the **Trajectory Prediction** task is defined as predicting the future positions $\hat{\mathbf{s}}_t, t \in \{1, \ldots, T\}$ for all agents in the scene over a fixed time horizon T. Following previous methods [1], [7], we predict a set of K trajectories $\hat{\mathbf{s}}_t^k,\ k \in \{1, \ldots, K\}$ with associated probabilities for each agent. We also predict 2D Gaussian uncertainties for each trajectory waypoint $\hat{s}_t^k$.

The **Occupancy Prediction** task is defined as predicting occupancy grids $\hat{O}_t,\ t \in \{1, \ldots, T\}$ with spatial dimensions $W \times H$. Each cell $\hat{O}_t(x, y)$ in the occupancy grid $\hat{O}_t$ contains a value in the range $[0, 1]$ representing the probability that any part of any agent box overlaps with that grid cell at time t. The ground-truth occupancy grids are constructed by rendering future agent boxes in BEV as binary maps. Since the planner reacts to different agent classes differently, we predict separate occupancy grids for each agent class.

B. Sparse Whole-Scene Input Representation

We use a whole-scene coordinate system centered on the AV's position at $t = 0$ (see Fig. 2). All the current and past agent states (including the AV's) are transformed to this fixed coordinate system. The model inputs consist of three sets of points $\mathcal{P} = \mathcal{P}^r \cup \mathcal{P}^l \cup \mathcal{P}^a$, each with associated feature vectors. Agent points $\mathcal{P}^a$ are constructed by uniformly sampling a fixed number of points from the interior of each agent box. The agent points from all input timesteps co-exist. Each agent point carries the state attributes mentioned in Sec. III-A, plus a one-hot encoding of time. The road element points $\mathcal{P}^r$ are sampled uniformly from the lines and curves of the road structure. Each road point encodes position and element type. Traffic light points $\mathcal{P}^l$ are placed at the end of the traffic lanes that they control. Their attributes include position, time, and traffic light state (color).

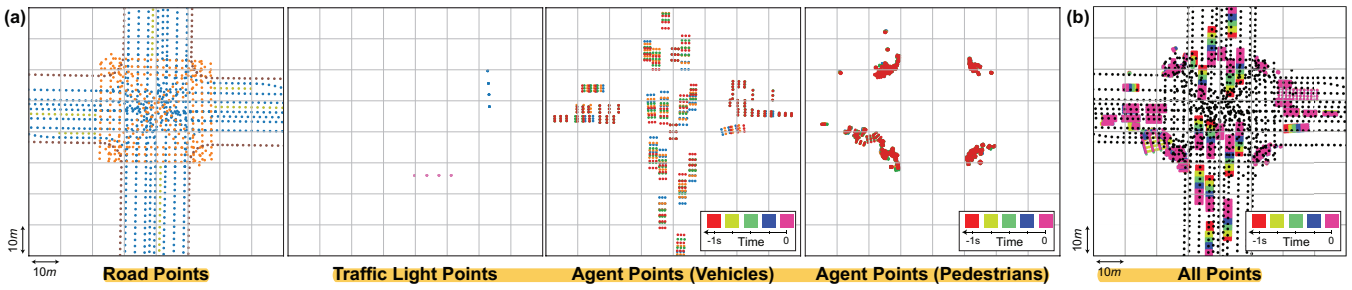

Fig. 2. Sparse Whole-Scene Input Representation. (a) Input point sets $\mathcal{P}^r$, $\mathcal{P}^l$ and $\mathcal{P}^a$ (vehicles and pedestrians) for an example scene. (b) All points.

C. Whole-Scene Encoder

Fig. 3 shows an overview of the StopNet architecture. It consists of an encoder, a ResNet backbone, and two heads for decoding trajectory and occupancy predictions from the shared scene features.

Inspired by PointPillars [13], the StopNet encoder discretizes the point set $\mathcal{P}$ into an evenly-spaced grid of $M \times N$ pillars in the x-y plane, $\{\pi_1, \pi_2, \ldots, \pi_{MN}\}$. The points in each pillar are then augmented with a tuple (x_c, y_c, x_{offset}, y_{offset}) where the c subscript denotes distance to the arithmetic mean of all points in the pillar and the *offset* subscript denotes the offset from the pillar center. We then apply a simplified version of PointNet [28] to encode and aggregate the features from all points in each pillar π_j. In particular, we apply a linear fully-connected (FC) layer followed by BatchNorm and a ReLU to encode each point. A max operation is then applied across all the points within each pillar to compute a single feature vector per pillar as

$$f_{\pi_j} = \text{MaxPool}\Big(\{\text{ReLU}(\text{BN}(\text{FC}(p_i)))\}_{p_i \in \pi_j}\Big). \quad (1)$$

The $M \times N$ feature map produced by the encoder is then processed through a ResNet backbone, reshaped to $W \times H$, and concatenated with binary occupancy grids rendered from the current positions of scene agents. The resulting feature map is then shared by a trajectory decoder and an occupancy grid decoder to produce the final predictions of the model.

D. Per-Agent Trajectory Decoder

To predict trajectories, we use the trajectory decoder architecture and losses from MultiPath [7]. The trajectory decoder extracts patches of size 11×11 centered on each agent location from the whole-scene features, thus operating on a per-agent basis. Note that while trajectory prediction head is agent-centric, the bulk of the model computation is whole-scene, and this dominates the overall processing time.

The trajectory decoder uses a fixed set of pre-clustered potential trajectories as an *anchor-set*, and ground-truth trajectories are assigned an anchor via closest Euclidean distance. For each anchor, the decoder regresses per-waypoint deltas from the anchor trajectory, yielding a Gaussian mixture at each timestep. The losses consist of a softmax cross-entropy classification loss over anchors $\mathcal{L}_s$, and within-anchor squared L_2-norm regression loss $\mathcal{L}_r$.

E. Occupancy Grid Decoder

The occupancy grid decoder processes the whole-scene feature map at once through a very lightweight CNN, which is repeated for each timestep t and produces occupancy logits for each class a as separate channels. The per-cell occupancy probabilities are obtained by applying a sigmoid function to the logits. The occupancy loss is defined as

$$\mathcal{L}_o(\hat{\mathcal{O}}, \mathcal{O}) = \frac{1}{WH} \sum_a \sum_t \sum_x \sum_y \mathcal{H}(\hat{\mathcal{O}}_t^a, \mathcal{O}_t^a), \quad (2)$$

where $\mathcal{H}$ denotes the cross-entropy function and $\mathcal{O}_t^a$ denotes the ground-truth occupancy for agent class a at time t.

F. Co-Training and Consistency Loss

In addition to co-training the trajectory and occupancy decoders, we find it useful to employ a consistency loss to encourage agreement between the per-agent trajectory predictions and whole-scene occupancy grids. The trajectory predictions with the highest predicted likelihood are rendered as oriented bounding boxes and aggregated by agent class as $\tilde{\mathcal{O}}_t^a$. Consistency with predicted occupancy outputs $\hat{\mathcal{O}}_t^a$ is then computed similarly to computing cross-entropy with the ground-truth as $\mathcal{L}_c(\hat{\mathcal{O}}, \tilde{\mathcal{O}}) = \mathcal{L}_o(\hat{\mathcal{O}}, \tilde{\mathcal{O}})$.

The loss function for the most general variant of our model is then summarized as

$$\mathcal{L} = \underbrace{\lambda_o \mathcal{L}_o}_{\text{Occupancy Loss}} + \overbrace{\lambda_s \mathcal{L}_s + \lambda_r \mathcal{L}_r}^{\text{Trajectory Loss}} + \underbrace{\lambda_c \mathcal{L}_c}_{\text{Consistency Loss}} \quad (3)$$

where λ_o, λ_s, λ_r, and λ_c are the respective loss weights.

IV. Experiments

A. Datasets

Crowds Dataset. This dataset is a revision of the Waymo Open Motion Dataset [29] focused on crowded scenes. It contains over 13 million scenarios spanning over 500 hours of real-world driving in several urban areas across the US. The scenarios contain dynamic agents, traffic lights and road network information. All scenarios contain at least 20 dynamic agents.

Interaction & Argoverse Datasets. We also evaluate our proposed method on the Interaction [2] and Argoverse [1] datasets. The Interaction dataset contains interactive driving scenarios involving multiple agents. In the Argoverse dataset,

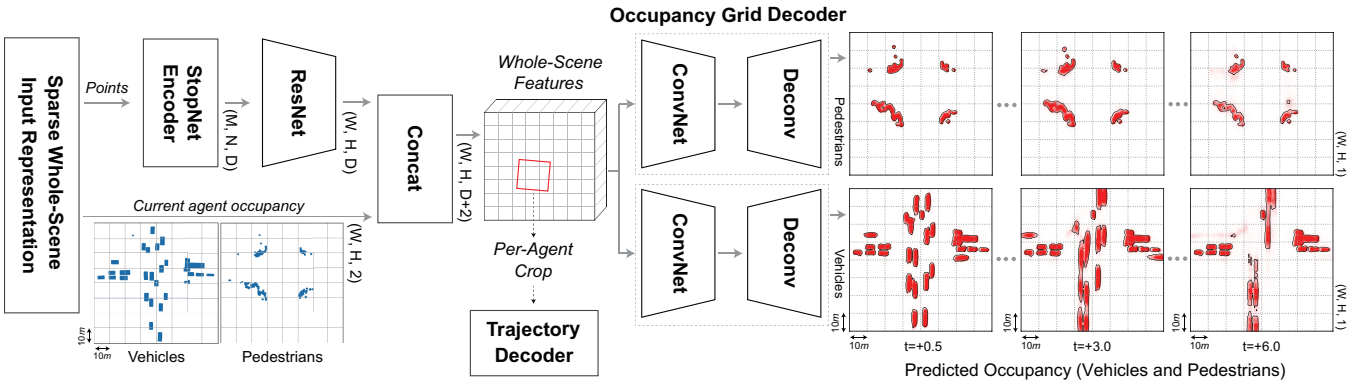

Fig. 3. An overview of the StopNet architecture. The encoder processes the input point set $\mathcal{P}$ and produces a feature map, which is used to predict both per-agent trajectories and whole-scene occupancy grids for each agent type. Input agent boxes at $t = 0$ are also rendered in BEV as binary features and fed to the trajectory and occupancy grid decoders.

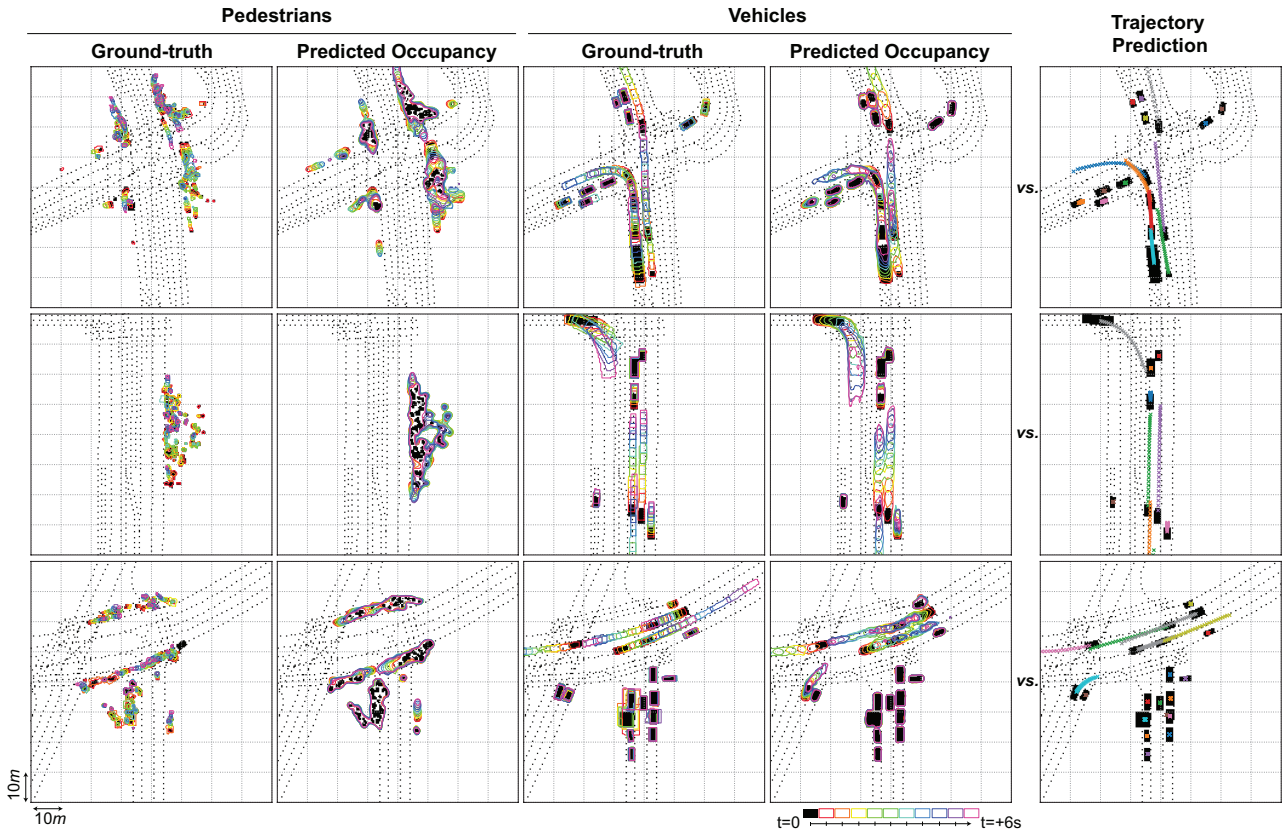

Fig. 4. Example occupancy and trajectory predictions from StopNet. **Left four columns:** Ground-truth and predicted occupancy grids are visualized through time as color-coded contour lines (from red for near future to purple for far future), where each contour contains values with probability > 0.5. **Right column:** For trajectories, different colors map to different agents. The dotted lines represent road points and the black boxes represent the current location of agents at $t = 0s$.

only one agent has future ground-truth, making it less interesting for our multi-agent whole-scene method. We, however, report scores on this dataset as well.

B. Training Setup

We train three variants of our model: M_T is trained only with a trajectory loss, M_O is trained only with an occupancy loss, and M_{TO}, which uses co-training and a consistency loss. All models are trained from scratch using an Adam optimizer [30], with a learning rate of 0.0004 and batch size of 8. We clip the gradient norms [31] above 0.1. The loss weights are $\lambda_o = 100.0$, $\lambda_s = 1.0$, $\lambda_r = 0.16$, and $\lambda_c = 10.0$, determined using light grid search. The input field of view is $160m \times 160m$, corresponding to an effective sensing range of $80m$ for the AV. Our encoder uses $M \times N = 80 \times 80$ pillars. We sample 8×8 input points uniformly from the interior of all agent boxes. Our occupancy decoder has a resolution of $W \times H = 400 \times 400$, predicting occupancy over $T = 10$ linearly-spaced timesteps up to 6 seconds in the future, *i. e.*, $t \in \{0.6, 1.2, \ldots, 6.0\}$. All figures show an

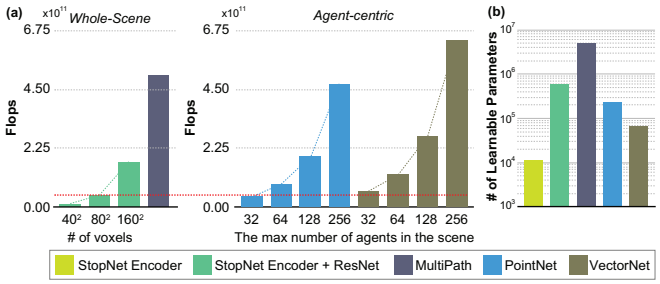

Fig. 5. Comparison of **(a)** number of flops, and **(b)** number of learnable parameters (log scale) for different model **encoders**. The dotted red line highlights the 80×80 pillars configuration used in our experiments reported in Table I.

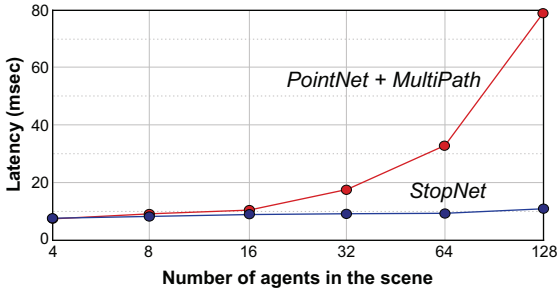

Fig. 6. StopNet scales well as the number of agents in the scene increase. For agent-centric models, latency grows linearly with the number of agents.

$80m \times 80m$ center crop of the output to show more details.

C. Metrics

Trajectory Metrics. We use two standard Euclidean-distance-based metrics [1]: the minimum average displacement error, $\text{minADE}_k = \min_k \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{s}_t - \hat{\mathbf{s}}_t^k||_2$, and minimum final displacement error, $\text{minFDE}_k = \min_k ||\mathbf{s}_T - \hat{\mathbf{s}}_T^k||_2$, where $\mathbf{s}$ denotes the ground-truth. We also report miss rate (MR), which measures the ratio of trajectories where none of the predictions are within $\{1, 2\}$ meters of the ground-truth according to FDE.

Occupancy Metrics. Evaluation metrics for occupancy grids in the context of motion forecasting have not been well documented in the existing literature. An intuitive choice, however, is the mean cross entropy (CE) error between the predicted occupancy grids $\hat{\mathcal{O}}_t^a$ and the ground-truth $\mathcal{O}_t^a$ as $\frac{1}{WH}\sum_{x,y}\mathcal{H}(\hat{\mathcal{O}}_t^a, \mathcal{O}_t^a)$. We also employ evaluation metrics commonly used for binary segmentation [32]: We use a linearly-spaced set of thresholds in $[0, 1]$ to compute pairs of precision and recall values to estimate the area under the PR-curve as AUC. We also measure the probabilistic area of overlap as Soft Intersection-over-Union [33]:

$$\text{Soft-IoU} = \sum_{x,y} \hat{\mathcal{O}}_t^a \mathcal{O}_t^a \Big/ \left(\sum_{x,y} \hat{\mathcal{O}}_t^a + \sum_{x,y} \mathcal{O}_t^a - \sum_{x,y} \hat{\mathcal{O}}_t^a \mathcal{O}_t^a \right) \quad (4)$$

D. Results

Trajectory Prediction. Table I compares our trajectory-only model M_T, and our co-trained model M_{TO} with state-of-the-art trajectory prediction models, MultiPath [7], TNT [10], and DESIRE [34] on three different datasets. To best evaluate the performance and latency characteristics of our encoder, we also compare our model with two agent-centric sparse encoders, namely VectorNet [9], and PointNet [28] as used by CBP [12]. For an even comparison, we couple these agent-centric encoders with the same trajectory decoder [7] we have adapted in our architecture. Following existing work [9], [12], we compute per-agent embeddings of the world and concatenate it with per-agent state embeddings before feeding it to the trajectory decoder.

TABLE I

TRAJECTORY PREDICTION PERFORMANCE ON DIFFERENT DATASETS. WE REPORT MODEL PERFORMANCE ON THE VALIDATION SET FOR INTERACTION AND ARGOVERSE DATASETS.

Crowds dataset	Sparse	Whole	$\text{minADE}_6 \downarrow$	$\text{minFDE}_6 \downarrow$	MR @ 1m, 2m
MultiPath [7]		✓	0.55	1.57	0.220, 0.385
VectorNet [9] + MultiPath [7]	✓		0.58	1.70	0.229, 0.399
PointNet [28] + MultiPath [7]	✓		0.53	1.60	0.235, 0.408
M_T (ours)	✓	✓	0.51	1.54	0.223, 0.400
M_{TO} (ours)	✓	✓	**0.51**	**1.49**	**0.215**, **0.384**
Interaction [2]	Sparse	Whole	$\text{minADE}_6 \downarrow$	$\text{minFDE}_6 \downarrow$	MR @ 1m, 2m ↓
DESIRE [34]		✓	0.32	0.88	-
TNT [10]	✓		0.21	0.67	-
VectorNet [9] + MultiPath [7]	✓		0.30	0.99	-
M_T (ours)	✓	✓	0.21	0.60	0.150, 0.018
M_{TO} (ours)	✓	✓	**0.20**	**0.58**	**0.136**, **0.015**
Argoverse [1]	Sparse	Whole	$\text{minADE}_6 \downarrow$	$\text{minFDE}_6 \downarrow$	MR @ 2m ↓
DESIRE [34]		✓	0.92	1.77	0.18
VectorNet [9] + MultiPath [7]	✓		**0.80**	1.68	**0.14**
M_T (ours)	✓	✓	0.87	1.68	0.19
M_{TO} (ours)	✓	✓	0.83	**1.54**	0.19

As Table I shows, our models match or exceed the performance of all the baselines, despite having a much smaller footprint. Note that the Argoverse dataset contains ground-truth future for a single agent, offering limited interactivity. These results show the advantage of our sparse whole-scene encoder over existing raster and agent-centric methods. Moreover, in all cases our co-trained model M_{TO} achieves the best performance on all trajectory metrics. This is likely due to the regularizing effect of unifying the two different output representation with a consistency loss.

Scalability. Fig. 5 compares the number of flops and learnable parameters used by the StopNet encoder vs. the whole-scene raster encoder from MultiPath and two agent-centric encoders. Including the ResNet backbone, our nominal encoder with 80×80 pillars uses about $1/10$ the number of flops used by MultiPath. Whole-scene approaches require a larger number of parameters as they need to have convolutional layers with a large receptive field. However, our core encoder uses much fewer parameters. Moreover, the compute required by our encoder is invariant to the number of agents—only a function of the pillar resolution. Sparse encoders, on the other hand, require linearly more compute with growing number of agents.

Fig. 6 shows the latency of our model (encoder + decoder) as a function of the number of agents, compared with an agent-centric model. The variable latency of agent-centric models poses a problem for coordination of processes run by the AV. Note that raster representations also require rendering

TABLE II

OCCUPANCY PREDICTION RESULTS USING RASTER / SPARSE INPUTS.

Input	# of pillars	CE ↓	Pedestrians				Vehicles			
			AUC ↑		IoU ↑		AUC ↑		IoU ↑	
			3s	6s	3s	6s	3s	6s	3s	6s
Raster	-	19.2	0.48	0.24	0.21	0.13	0.84	**0.73**	0.49	0.36
Sparse	20×20	18.6	0.48	0.24	0.22	0.13	0.83	0.71	0.50	0.35
Sparse	40×40	17.6	0.54	0.26	0.25	0.14	0.86	0.72	0.50	0.36
Sparse	80×80	17.2	0.56	0.27	**0.27**	**0.15**	**0.87**	**0.73**	**0.53**	**0.37**
Sparse	160×160	**17.0**	**0.59**	**0.28**	**0.27**	**0.15**	0.86	**0.73**	**0.53**	**0.37**

the model inputs, further increasing the effective latency.

Occupancy Prediction. Table II shows occupancy prediction results on the Crowds dataset. To evaluate our sparse input representation, we also train baseline models using BEV raster inputs. Following existing work [8], [6], [7], [21], we render road structure, speed limits, traffic lights, and agent history at 400×400 resolution and feed the stacked images to the model. We also ablate the pillar resolution for our sparse encoder. Results reflect the advantage of our sparse scene representation. While 160×160 pillars work best, 80×80 pillars have comparable performance at lower complexity.

Occupancy Grids vs. Trajectories. Occupancy grid and trajectory representations have complementary advantages, which motivates StopNet to support both output formats. Trajectory models often output tens of potential trajectories per agent, which need to be taken into consideration as constraints in the planning algorithm. The size of the trajectory outputs grows linearly with the number of agents in the scene, while the number of potential agent interactions grows quadratically. This variability makes it challenging to complete planning for the AV under a fixed compute budget. Occupancy grids require fixed compute to generate and consume regardless of the number agents in the scene. They also capture the full extents of agent bodies, as opposed to just center locations, which simplifies calculating overlap probabilities. On the other hand, trajectory sets can be represented as sparse sequences, which are more compact. In scenes with few agents, processing few trajectories can be done faster than processing a dense probability map.

Fig. 4 shows occupancy and trajectory predictions by our model on three sample urban driving scenes. We observe that our occupancy representation is especially effective in situations where occupancy blobs can capture the collective behavior of groups, and eliminate the need for generating trajectory sets for individual agents. The occupancy representation is particularly useful in busy urban scenes, where trajectory prediction models face challenges caused by noisy detection and poor tracking due to occlusions.

Because of the different representations, it is difficult to directly compare the quality of trajectories with occupancy grids. As a proxy, we convert predicted trajectories to occupancy by rendering agent boxes on locations predicted by the trajectory waypoints. Since the model predicts multiple trajectories, we render each agent box with an intensity

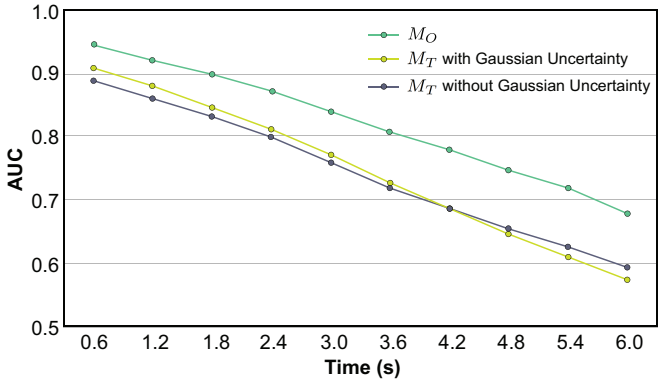

Fig. 7. Comparison of our occupancy model M_O with two versions of our trajectory model M_T trained with and without Gaussian uncertainties on occupancy prediction for vehicles. While M_O predicts occupancy directly, the top six trajectory outputs from M_T models have been converted (rendered) into an occupancy grid representation. Results show that the rich non-parametric representation is more suitable for occupancy prediction.

matching the associated probability for the corresponding trajectory. Fig. 7 shows a comparison between our native occupancy model M_O and occupancies converted from our trajectory model M_T. We train two versions of M_T, once with and once without Gaussian uncertainties. The two-dimensional variance of each Gaussian is factored in by first rendering the probability density function of the Gaussian and then convolving that with the rendered agent box. As the plot shows, M_T underperforms M_O on this metric, which serves as validation for the utility of occupancy grids. Moreover, the plot shows that while including Gaussian uncertainties helps M_T in the near future, it hurts performance over longer prediction horizons. The position uncertainty of road agents is often more complex than a Gaussian mixture model, and is best represented with the rich non-parametric distributions supported by occupancy grids.

V. CONCLUSIONS

In this paper, we proposed StopNet, a novel, efficient, and scalable motion forecasting method that accommodates sparse inputs in a whole-scene modeling framework, and co-trains trajectory and occupancy representations. Our model has an almost fixed compute budget and latency, independent of the number of agents in the scene. Likewise, our occupancy predictions can be consumed with fixed compute in a planning algorithm. In addition to this higher efficiency and scalability, our experiments show that our model matches or outperforms performance of prior methods under standard trajectory and occupancy metrics. In future work, it would be interesting to extend the occupancy representation with per-pixel motion information, enabling the model to trace predicted occupancies back to the original agents. Future research could explore applications of StopNet to reasoning about occupancy of occluded objects—a challenging task for pure trajectory-based representations.

REFERENCES

[1] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *CVPR*, 2019.

[2] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," *arXiv:1910.03088*, 2019.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020, pp. 11 621–11 631.

[4] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," https://level5.lyft.com/dataset/, 2020.

[5] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," in *CoRL*, 2018, pp. 947–956.

[6] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *CVPR*, 2019, pp. 8454–8462.

[7] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *CoRL*, 2019.

[8] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *RSS*, 2019.

[9] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *CVPR*, 2020, pp. 11 525–11 533.

[10] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov, "Tnt: Target-driven trajectory prediction," *CoRL*, 2020.

[11] S. Khandelwal, W. Qi, J. Singh, A. Hartnett, and D. Ramanan, "What-if motion prediction for autonomous driving," *arXiv preprint arXiv:2008.10587*, 2020.

[12] E. Tolstaya, R. Mahjourian, C. Downey, B. Vadarajan, B. Sapp, and D. Anguelov, "Identifying driver interactions via conditional behavior prediction," *ICRA*, 2021.

[13] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *CVPR*, 2019, pp. 12 697–12 705.

[14] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proceedings of the National Conference on Artificial Intelligence*, 1996, pp. 944–951.

[15] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[16] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *NeurIPS*, 2019, pp. 15 424–15 434.

[17] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *ECCV*, 2018, pp. 772–788.

[18] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-head attention for multi-modal joint vehicle motion forecasting," in *ICRA*, 2020, pp. 9638–9644.

[19] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *CVPR*, 2019, pp. 12 126–12 134.

[20] N. Sriram, B. Liu, F. Pittaluga, and M. Chandraker, "Smart: Simultaneous multi-agent recurrent trajectory prediction," in *ECCV*. Springer, 2020, pp. 463–479.

[21] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *CVPR*, 2020, pp. 14 074–14 083.

[22] Y. Biktairov, M. Stebelev, I. Rudenko, O. Shliazhko, and B. Yangel, "Prank: motion prediction based on ranking," *NeurIPS*, vol. 33, 2020.

[23] T. Buhet, E. Wirbel, and X. Perrotton, "Plop: Probabilistic polynomial objects trajectory planning for autonomous driving," *CoRL*, 2020.

[24] A. Jain, S. Casas, R. Liao, Y. Xiong, S. Feng, S. Segal, and R. Urtasun, "Discrete residual flow for probabilistic pedestrian behavior prediction," in *CoRL*. PMLR, 2020, pp. 407–419.

[25] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F.-C. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu, *et al.*, "Multixnet: Multiclass multistage multimodal motion prediction," *IV*, 2020.

[26] J. Strohbeck, V. Belagiannis, J. Müller, M. Schreiber, M. Herrmann, D. Wolf, and M. Buchholz, "Multiple trajectory prediction with deep temporal and spatial convolutional neural networks," 2020.

[27] S. Casas, A. Sadat, and R. Urtasun, "Mp3: A unified model to map, perceive, predict and plan," *CVPR*, 2021.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017, pp. 652–660.

[29] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," *ICCV*, 2021.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[31] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013, pp. 1310–1318.

[32] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *TPAMI*, vol. 40, no. 4, pp. 834–848, 2017.

[33] G. Máttyus, W. Luo, and R. Urtasun, "Deeproadmapper: Extracting road topology from aerial images," in *ICCV*, 2017, pp. 3438–3446.

[34] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *CVPR*, 2017, pp. 336–345.