Simulating Behaviors of Traffic Agents for Autonomous Driving via Interactive Autoregression

Xiaoyu Mo, Haochen Liu, Zhiyu Huang, and Chen Lv.

***Abstract*— In this report, we present our solution for the Waymo Open Dataset Sim Agents Challenge (WOSAC). Our method aims to address the driving scenario by employing a three-stage approach. Firstly, we generate joint goal sets for the agents involved in the scenario, followed by the assignment of target centerlines (TCLs) to each agent based on their predicted goals. In the second stage, we introduce a random sampling process to assign a TCL to one of the agents, ensuring diverse simulations. Since the joint goal sets predicted by the motion prediction method might not align with the road structure, we further refine the goals for each agent by re-estimating them according to their assigned TCLs. These re-estimated goals serve as inputs for our interactive auto-regressive goal-directed planner (GDP), which serves as the third stage of our method that generates rollouts for simulation.**

***Index Terms*—Traffic simulation, connected vehicles, graph neural networks, heterogeneous interactions.**

I. Related Work

A. Map-Adaptive Trajectory Prediction

Map-adaptive methods can well generalize to handle different lane topologies, such as intersections, roundabouts, and other unusual road structures [1]. GoalNet [1] represents the inputs and outputs in a path-relative coordinate frame and proposes to use a GNN to generate a variable number of trajectories according to the number of available paths of the target vehicle. However, GoalNet does not consider vehicles behind the target or those in other lanes, which can significantly impact the target vehicle's behavior. Following GoalNet, authors of [2] propose an integrated framework for map-adaptive multimodal trajectory prediction. They represent the driving scene as a hierarchical heterogeneous graph containing both agents and their candidate centerlines (CCLs) and propose a hierarchical graph operator (HGO) to handle the heterogeneity. They introduce a virtual node to contain the target vehicle's dynamics feature for non-map-adaptive behavior prediction. Incorporating the virtual node into the graph, they can generate CCL-following, scene-reasoning, and motion-maintaining predictions simultaneously.

B. Data-Driven Traffic Simulation

Data-driven traffic simulation can be built on trajectory prediction methods. For example, TrafficSim [3] extends the motion prediction method ILVM [4] and realizes traffic simulation in an auto-regressive manner. TrafficBots [5] introduces Transformers [6] into a VectorNet [7] for map encoding and formulates the traffic simulation task as a world model [8] tailored for the planning module for autonomous vehicles. We extend the map-adaptive trajectory prediction method proposed in [2] and follow the requirements of WOSAC [9] to simulate traffic agents in an auto-regressive manner.

Xiaoyu Mo, Haochen Liu, Zhiyu Huang, and Chen Lv are with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, 639798, Singapore. (e-mail: xiaoyu006@e.ntu.edu.sg, haochen002@e.ntu.edu.sg, zhiyu001@e.ntu.edu.sg, lyuchen@ntu.edu.sg).

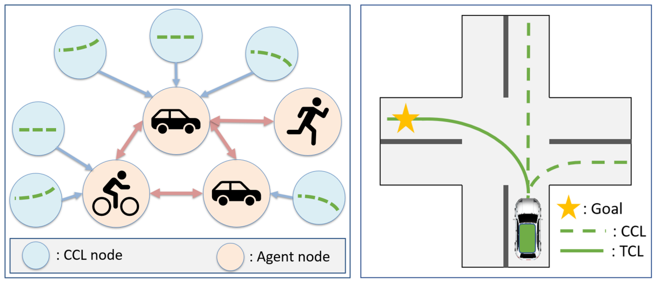

Fig. 1. *Left*, an illustration of the hierarchical heterogeneous scene graph. *Right*, CCL v.s. TCL. An agent's TCL is selected from its CCLs according to its ground truth goal point. In inference, the TCL can be manually assigned by users to generate complex interactions.

II. Method

Our model for the Sim Agents challenge consists of three sub-models, namely 1) the multi-agent goal predictor (MAG), 2) the target centerline-based goal estimator (TGE), and 3) the interactive auto-regressive goal-directed planner (GDP). The proposed model realizes multi-agent traffic simulation in three stages for each update: 1) assign a target centerline TCLs to each agent in the scene according to their goals predicted by the multi-agent goal predictor, 2) re-estimating their goals according to their TCLs separately, 3) apply GDP to generate the final trajectories for agents to follow.

A. Problem Formulation

In the Waymo Open Sim Agents Challenge (WOSAC), the driving task is formulated as a Hidden Markov Model, requiring participants to construct a "world model" for traffic simulation. This world model should exhibit autoregressive behavior for a specified number of steps (80 steps for 8 seconds) and be factorized into self-driving car (SDC) and non-SDC components [9]. Given the scene context which includes a one-second history of the states of traffic participants and signals and the local map, the task of WOSAC is to generate interactive motions for both SDC and non-SDCs in an autoregressive manner. The simulation will be executed for a duration of 8 seconds into the future, with a sampling frequency of 10 Hz.

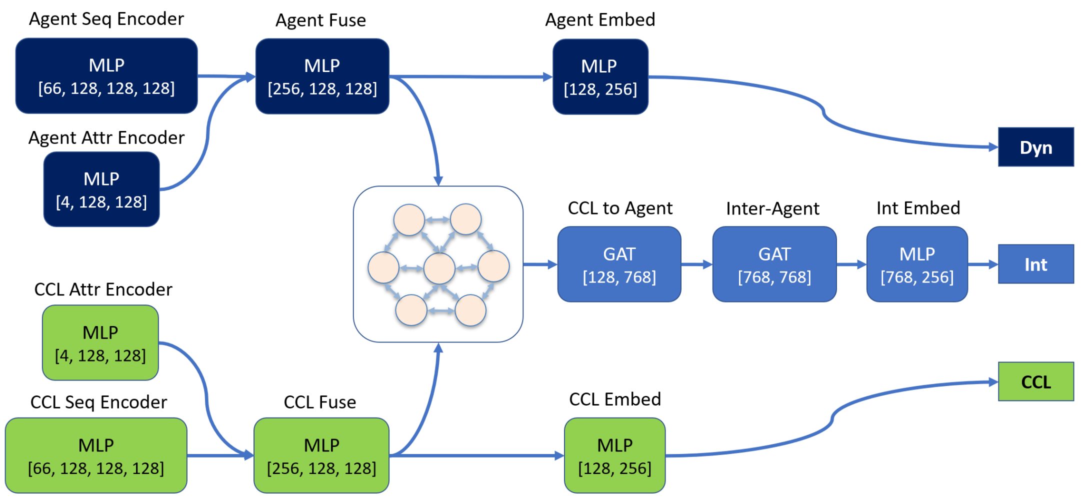

Fig. 2. **Detailed structure of the scene graph encoding module.** Given the sequences (Seq) and attributes (Attr) of Agents/CCLs, we apply corresponding encoders to them and then fuse them to obtain individual representations of both agents and CCLs. These representations are then used as node features in the scene graph, which is processed by the hierarchical graph operator (HGO [2]) with two graph attention networks (GATs) for interaction modeling. Finally, we embed all representations to the same dimension and obtain the agents' dynamics (Dyn), interaction (Int) features, and CCL features for the downstream modules.

At each time step, participants are required to simulate the 3D positions and headings (x, y, z, h) of all the traffic participants (valid agents in the WOSAC dataset).

B. Scene Representation and Encoding

To establish the relationship between the scene representations of traffic participants (A_t) and their candidate centerlines (CCLs) (C_t), we utilize a heterogeneous scene graph. Each node in the graph represents either an agent (vehicles, cyclists, and pedestrians) or a CCL of an agent. Agent nodes in the graph store the past states of the corresponding agent over the last second. These states include relevant measurements (position, velocity, and orientation). Additionally, each agent node is assigned attributes, including shape (length, width, and height) and type (vehicles, cyclists, and pedestrians), to provide further contextual information. CCL nodes in the graph contain the waypoints that define the optional centerlines for the associated agent. The attributes of a CCL node encompass details such as traffic light states, the speed limit for the CCL, and indicators indicating the presence of stop signs or speed bumps along the CCL.

We place all the agents and CCLs in an ego-centric coordinate framework, where ego is the self-driving car (SDC), making the origin at the SDC's current position and the horizontal axis align with its current heading direction. All agents' past states and their CCLs are represented by sequences in this frame of reference. The CCLs of an agent at time t are obtained from the road graph by applying a depth-first search on the road graph from its current lane. Every agent is connected to its neighbors (within a distance of 50 meters in our setting) and CCLs. There is no link between an agent's CCL to another agent. So the driving scene is represented by a hierarchical heterogeneous graph, where the lower-level graph is an agent-CCL graph (shown in blue arrows in the left part of Fig. 1) containing an agent and its CCLs, and the higher-level graph is an interaction graph (shown in red arrows in the left part of Fig. 1) containing all agents in the scene.

Rather than using GRUs for both Agents (*agn*) and CCLs (*ccl*) sequences, we use multi-layer perceptions (MLPs) to speed up training and inference. In addition to sequential (*seq*) information, we also encode their attributes (*atr*) and concatenate them with their sequential features to construct the node features in the scene graph.

$$d_t^i = \mathrm{MLP}_{agn}^{seq}(p_t^i), i \in \mathcal{A}, \tag{1}$$

$$a_t^i = \mathrm{MLP}_{agn}^{atr}(z^i), i \in \mathcal{A}, \tag{2}$$

$$q_t^j = \mathrm{MLP}_{ccl}^{seq}(c_t^j), j \in \mathcal{C}, \tag{3}$$

$$b_t^j = \mathrm{MLP}_{ccl}^{atr}(m_t^j), j \in \mathcal{C}, \tag{4}$$

where $\mathcal{A}$ and $\mathcal{C}$ contain the indices of agent and CCL nodes in the graph, p_t^i is the past states of an agent i, z^i is its attributes, c_t^j is the sequence of a CCL j, and m_t^j is its attributes. Applying the above MLP encoders accordingly to all the nodes, we obtain agents' dynamics and attribute features (d_t^i and a_t^i) and CCLs' sequential and attribute features (q_t^j and b_t^j). For an agent node, the node feature is $[d_t^i, a_t^i]$, where $[\cdot|\cdot]$ means concatenation, while for a CCL node, the feature is $[q_t^j, b_t^j]$.

We use a two-layer Hierarchical Graph Operator (HGO) [2] to process the scene graph to obtain a feature vector that summarises all information in the scene for the simulation. The first stage of our HGO is for each agent's CCL awareness. By masking out inter-agent edges, we allow all agents to collect information about their CCLs. Then in the second stage, we allow agents to gather information from their neighborhood to make all the agents interaction-aware. For details about the HGO and edge-masking technique, please refer to [2]. The scene encoder takes A_t and C_t as inputs and outputs all agents' dynamics (S_D), interaction (S_I), and CCL (S_C) features:

$$S_D, S_I, S_C = \mathrm{ENC}_{\mathrm{scn}}(A_t, C_t), \tag{5}$$

where $\mathrm{ENC}_{\mathrm{scn}}$ is the scene (scn) encoder (ENC). Fig. 2 shows the structure of the scene encoding stage, where all three kinds of features are embedded into a fixed size for the next decoding stage.

C. Multi-Agent Goal Prediction (MAG) and TCL assignment

In this stage, we want to generate joint goal sets for all agents in the scene, then assign a TCL (selected from an agent's CCLs) per agent given a joint goal set. Based on the interaction and dynamics features of all the agents, our model first predicts four joint goal sets ($\hat{G} = \{\hat{G}_1, \hat{G}_2, \hat{G}_3, \hat{G}_4\}$) for them simultaneously to capture the multimodality of real-world driving. For an agent i:

$$\hat{G}^i = \mathrm{DEC}_{MAG}([S_I^i, S_D^i]), \tag{6}$$

where $[S_I^i, S_D^i]$ is the concatenation of its interaction and dynamics feature, $\hat{G}^i$ is the predicted multimodal goal of agent i, and DEC_{MAG} is a 2-layer MLP. Given a predicted goal set (e.g., $\hat{G}_1$) we assign a TCL for each agent from their CCLs according to a distance criterion (the TCL is the CCL that is closest to the agent's predicted goal while within the distance threshold, see the right part of Fig. 1 for an illustration.). If none of an agent's CCLs satisfy the criteria, we will assign a fake TCL to it. In this stage, we initialize a TCL for every agent.

We train the MAG predictor via a joint multimodal loss inspired by the MTPLoss [10] to force the model to predict joint goal sets. For each scenario, we compute the joint displacement error (JDE) for each modality (we output 4 joint goal sets for each scenario) and use the minimum JDE over 4 joint predictions as the loss for backpropagation. This is similar to the regression loss proposed in [10]. Specifically, the displacement error between two points $\hat{p} = (\hat{x}, \hat{y})$ and $p = (x, y)$ can be calculated by

$$DE(\hat{p}, p) = \sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2}. \tag{7}$$

Then for each goal set $\hat{G}_m$ containing goals of N agents in the scene, we can calculate its joint displacement error (JDE) as

$$JDE_m = \frac{1}{N} \sum_{i=1}^{N} DE(\hat{G}_m^i, G^i), \tag{8}$$

where $\hat{G}_m^i$ is the predicted goal of agent i in this joint goal set, and G^i is the ground truth. Then the best joint goal set $\hat{G}_{m^*}^i$ is the one with the minimum JDE:

$$m^* = \underset{m \in \{1,2,3,4\}}{\arg\min}\ JDE_m. \tag{9}$$

Then we use JDE_{m^*} as the loss function for backpropagation.

By conditioning the behavior of a traffic participant on a TCL, our method allows manually assigning a TCL to generate counterfactual behaviors of one or more agents. Users can force an agent to interact strongly with the SDC in training for corner case simulation. For the WOSAC challenge, we randomly assign a TCL to a sampled agent for diverse simulations. Since a TCL is chosen from the agent's CCLs and its CCLs are obtained via a depth-first search on the road graph, the assigned TCL is compliant with the road structure. Given a predicted goal set (e.g., $\hat{G}_1$), we do random TCL assignments eight times, so that we can generate 32 TCL sets to meet the challenge's requirements (32 rollouts for a given scenario).

D. TCL-based Goal Estimation

Given the driving scene and a TCL of an agent, the *TCL-based goal estimator (TGE)* wants to predict where the agent is most likely to be in the next eight seconds (the predefined simulation horizon). Different from the multi-agent goal prediction (MAG) module, which outputs multiple joint goal sets for all agents in the scene, TGE focuses on a single agent and estimates a deterministic point as its goal conditioned on a TCL. We assume that an agent's goal is conditioned on three factors: 1) its interaction feature s_I^i, 2) its dynamics feature s_D^i, and 3) the TCL feature $s_C^{i,\tau}$. Rather than sampling positions, either densely or sparsely [11], [12], we propose to directly estimate the goal position by assuming that the goal distribution of a vehicle is unimodal given the above factors. Then the goal estimator is formulated as:

$$\hat{G}^\tau = \mathrm{DEC}_{TGE}(S_I, S_D, S_C^\tau), \tag{10}$$

where DEC_{TGE} is implemented with an MLP and $\hat{G}^\tau$ is the predicted goal set according to a TCL set for all agents in the scenario. In training time, we feed the ground truth TCL into this goal estimator so that the learning process is not affected by the selection of TCL.

We use the MSE loss between the predicted goal position and the ground truth to train the goal estimator for simplicity, but other loss functions, like Huber loss, can also be used.

E. Goal-Directed Planning

Given a goal set ($\hat{G}^\tau$) generated by the TGE (Eq. 10), the goal-directed planner (GDP) rolls out a traffic simulation in an auto-regressive manner for both SDC and the world agents separately while considering the interaction among them. The auto-regression is implemented via GRU Cells. At each time step, the SDC updates its states considering both its hidden feature and the new input (i.e., states of the world agents). All the world agents (*wld*, non-SDC agents) will be able to react to the SDC agent while trying to accomplish their driving goals. For the first step, we use $x_0^{ego} = [S_I^{ego}, S_D^{ego}, S_G^{ego}]$ as the input to the SDC's GRU network, and $x_0^{wld} = [S_I^{wld}, S_D^{wld}, S_G^{wld}]$ as the input to the world agent's GRU. S_G is the embedding of goals. Both GRU cells have a hidden size of 384. Their hidden states h_0^{ego}, h_0^{wld} are initialized randomly:

$$h_1^{ego} = \mathrm{GRU}(x_0^{ego}, h_0^{ego}), \tag{11}$$

$$h_1^{wld} = \mathrm{GRU}(x_0^{wld}, h_0^{wld}). \tag{12}$$

From step 2 onward, we use the input from hidden states of the world agents with a max operation over the feature dimension and use the input from hidden states of the SDC agents for each world agent:

$$h_t^{ego} = \mathrm{GRU}(\max(h_{t-1}^{wld}), h_{t-1}^{ego}), t \in [2, 80], \tag{13}$$

TABLE I
RESULTS ON THE WOSAC LEADERBOARD

Method Name	Realism Meta Metric	Kinematic Metrics	Interactive Metrics	minADE
MVTE	0.5168	0.4202	0.5289	1.6770
MTR+++	0.5091	0.4175	0.5186	1.8698
CAD	0.4321	0.3357	0.4355	2.3146
multipath	0.4240	0.1792	0.5328	2.0517
Ours	0.3941	0.3574	0.3923	3.6198
QCNeXt	0.3920	0.3109	0.4454	1.0830
sim_agents_tutorial	0.3201	0.1384	0.4294	3.1080
linear_extrapolation_baseline_tutorial	0.2576	0.0735	0.3548	7.5148

$$h_t^{wld} = \text{GRU}(h_{t-1}^{sdc}, h_{t-1}^{wld}), t \in [2, 80]. \tag{14}$$

In this setting, the SDC agent operates according to its own states and features of all the world agents, and our world agents are reactive to the SDC agents since their motions are a function of both their past hidden states and that of the SDC agent.

Then, the hidden states for each agent (*ego* and *wld*) over the prediction horizon (80 steps, 8 seconds into the future) are decoded using another MLP to 4-dimensional vectors representing its states (x, y, z, heading) at each step. For a scenario containing N agents, the output tensor for an 8-second roll-out is a tensor of shape $[N \times 80 \times 4]$.

We train the goal-directed planner via a Smooth L1 Loss as implemented by PyTorch.

III. Training and Simulation

A. Dataset

We train and evaluate our model on the training split of the Waymo open motion dataset (WOMD) [13] and test it on the test split of Sim Agents Challenges 2023 [9].

B. Training

We split the whole WOMD dataset for training (90%) and in-process validation (10%). We use the AdamW optimizer with the weight decay set to 0.01 and the initial learning rate of 0.0001 to train the models for 30 epochs on a 2080Ti GPU. The learning rate gets half after epochs 20, 22, 24, 26, and 28. For both the MAG and TGE models, the batch size was set to 256, while for the GDP, the batch size is set to 64. We use both the SDC and target agents defined by WOMD for training and generate rollouts of all valid agents for simulation. Note that the three modules (i.e., MAG, TGE, GDP) are trained separately. Both TGE and GDP are unimodal (deterministic) methods, while they can generate diverse simulations by conditioning on different TCLs and Goals.

C. Simulation

A simulation is realized by applying these three modules in a cascading manner. Given a scenario, we first apply the MAG model to generate 4 joint goal sets. Then, for each goal set, we initialize a TCL for each agent in the scene according to their predicted goal. To generate 8 simulations from each goal set, we randomly sample an agent, which has more than one CCLs and randomly assign a TCL for it. This stage allows us to manipulate the behavior of a world agent to enforce strong interaction with the SDC agent for specific testing purposes, while for the challenge, we just randomly select an agent to assign its TCL. After a TCL is assigned per agent, we can estimate the goal for each agent and then apply the GDP model to roll out a simulation. By iterating through all the goal sets, we can generate 32 different simulations as required by the challenge.

IV. Results

We show the results for the WOSAC in Tab. I. Where our model outperforms three submissions in terms of the Realism Meta metric, including the linear extrapolation baseline provided by Waymo and a generalization of the QCNet (QCNeXt for WOSAC), which was the winner of the Argoverse motion prediction challenge 2022. For details of these metrics for the challenge, please refer to [9]. Considering that WOSAC has a close relationship with the motion prediction task, the generalization of a challenge wining method can be regarded as a strong baseline, and outperforming QCNeXt demonstrates the effectiveness of our method for the Sim Agents challenge. We do not have further information regarding other methods submitted to the leaderboard. We look forward to knowing more about these methods to compare thoroughly and improve the performance of our method in the future. We show six sampled roll-outs in Fig. 3 for six different scenarios. It can be seen that the proposed method is able to generate traffic roll-outs for a variable number of traffic agents in different driving scenarios.

V. Conclusion

This report provides a comprehensive overview of our solution to the Sim Agents challenge, which focuses on enabling the autoregressive simulation of future interactions among traffic participants. Additionally, our solution offers the flexibility for users to customize the behaviors of world agents through target centerline (TCL) assignment.

References

[1] L. Zhang, P.-H. Su, J. Hoang, G. C. Haynes, and M. Marchetti-Bowick, "Map-adaptive goal-based trajectory prediction," *arXiv preprint arXiv:2009.04450*, 2020.

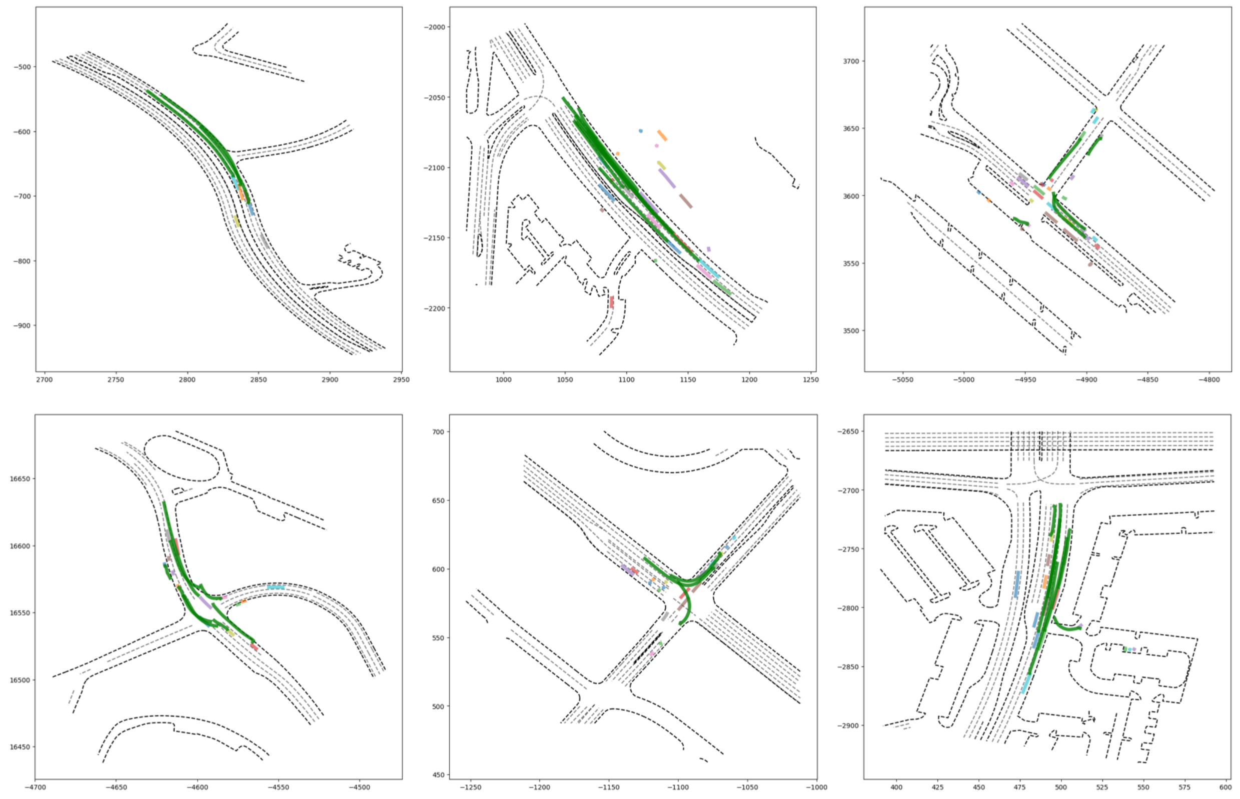

Fig. 3. **Visualization of the generated roll-outs in different scenarios.** The green lines show the roll-outs for agents of interest (tracks to predict in WOSAC) while the other lines show the past tracks of all valid agents in this scenario.

[2] X. Mo, Y. Xing, H. Liu, and C. Lv, "Map-adaptive multimodal trajectory prediction using hierarchical graph neural networks," *IEEE Robotics and Automation Letters*, 2023.

[3] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.

[4] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, "Implicit latent variable model for scene-consistent motion forecasting," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 624–641.

[5] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "Trafficbots: Towards world models for autonomous driving simulation and motion prediction," *arXiv preprint arXiv:2303.04116*, 2023.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[7] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[8] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," *Advances in neural information processing systems*, vol. 31, 2018.

[9] N. Montali, J. Lambert, P. Mougin, A. Kuefler, N. Rhinehart, M. Li, C. Gulino, T. Emrich, Z. Yang, S. Whiteson *et al.*, "The waymo open sim agents challenge," *arXiv preprint arXiv:2305.12032*, 2023.

[10] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2090–2096.

[11] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 303–15 312.

[12] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," *arXiv preprint arXiv:2109.01827*, 2021.

[13] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9710–9719.