DragTraffic: Interactive and Controllable Traffic Scene Generation for Autonomous Driving

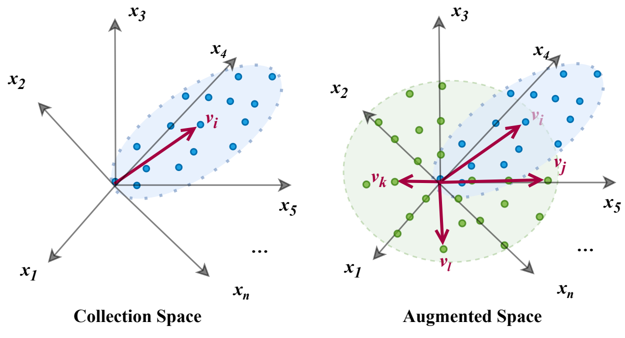
Sheng WANG[1], Ge SUN[1], Fulong MA[2], Tianshuai HU[1], Qiang QIN[3], Yongkang SONG[4], Lei ZHU[2] and Junwei LIANG[2]

***Abstract*— Evaluating and training autonomous driving systems require diverse and scalable corner cases. However, most existing scene generation methods lack controllability, accuracy, and versatility, resulting in unsatisfactory generation results. Inspired by DragGAN in image generation, we propose DragTraffic, a generalized, interactive, and controllable traffic scene generation framework based on conditional diffusion. DragTraffic enables non-experts to generate a variety of realistic driving scenarios for different types of traffic agents through an adaptive mixture expert architecture. We employ a regression model to provide a general initial solution and a refinement process based on the conditional diffusion model to ensure diversity. User-customized context is introduced through cross-attention to ensure high controllability. Experiments on a real-world driving dataset show that DragTraffic outperforms existing methods in terms of authenticity, diversity, and freedom. Demo videos and code are available at https://chantsss.github.io/Dragtraffic/.**

I. Introduction

The safety of autonomous driving systems relies heavily on the richness of the dataset scenarios. However, due to various constraints such as safety issues, geographical environment, and weather changes, it is difficult for collected data to cover all situations. This poses challenges for training and evaluating planning and prediction modules, especially for extreme scenarios. To address this, simulators such as SUMO [1] and CARLA [2] have been used to manually set scenarios. While rule-based simulations offer interpretability and viable trajectories without extensive data, they have limitations in accuracy, generalization, and adaptability. Furthermore, they require substantial expert knowledge to establish the necessary rules.

In contrast, data-driven methods have been developed to enable agents in these environments to emulate the behaviors of real traffic participants. The recent Sim Agents Challenge [3] based on the Waymo dataset provides a standard benchmark and specifies input and output forms for scene generation tasks. Several works have used learning-based methods to achieve good results, particularly in terms of accuracy [4] [5] [6]. However, most of these works formulate the scene generation task as a motion prediction task, requiring 10 frames of historical information as input and limiting the freedom of scene construction. As a result, they can only reason about future scenarios based on existing historical trajectories, while ignoring requirements such as scene editing and agent insertion. Other researchers have looked at generating challenging scenes in a more flexible way, such as SceneGen [7] and TrafficGen [8], which propose building scenes in two stages: vehicle placement and trajectory generation. However, a main shortcoming of these methods is the lack of controllability, which means they cannot ensure expected behavior. This serious problem leads to the generation process being directionless and extremely inefficient when the sample space is large, as shown in the Figure 1. The ideal sample data distribution should be distributed in all dimensions. However, due to factors such as cost and security, high-value data is often scarce. Another problem is that they only focus on vehicles and



$$\{v_1,\dots,v_i,\dots\} \subset \{v_1,\dots,v_i,v_j,v_k,v_l,\dots\} \sim \mathcal{R}^n$$

Fig. 1: **The dataset sample space.** The left image illustrates the distribution of the collected data, while the right image shows the expanded sample space achieved through data augmentation. In this context, x represents the different dimensions that constitute the dataset, and $\boldsymbol{v}$ represents the specific samples collected.

This work was supported by Lotus Technology Ltd. through The Hong Kong University of Science and Technology (GZ) under Cooperation Project R00082, Corresponding author: Junwei LIANG.

[1]Sheng WANG, Ge SUN and Tianshuai HU are with Robotics and Autonomous Systems, Division of Emerging Interdisciplinary Areas (EMIA) under Interdisciplinary Programs Office (IPO), The Hong Kong University of Science and Technology, Hong Kong SAR, China. {swangei, gsunah, thuaj}@connect.ust.hk

[2]Junwei LIANG is with Artificial Intelligence Thrust, Fulong MA and Lei ZHU are with Robotics and Autonomous Systems Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511400, China. {junweiliang, leizhu}@hkust-gz.edu.cn, fmaaf@connect.hkust-gz.edu.cn

[3]Qiang QIN is with Department of Production Engineering, KTH Royal Institute of Technology, Sweden. qiangq@kth.se

[4]Yongkang Song is with Lotus Technology Ltd, China. yongkang.song@lotuscars.com.cn

ignore other types of traffic participants, even though these participants have many interactions. Meanwhile, generative models have been well developed in contentmade great progress in sequence generation tasks such as text, pictures, and videos [9] [10] [11] [12] [13]. Some prior works are inspired of using generative models to create traffic scenes such as [14], [15] utilizing generative adversarial networks (GAN) to generate multiple trajectories for traffic agents. In order to obtain a more operational trajectory generation method, some researchers try to use conditional diffusion models such as motiondiffuser [16], and SceneDM [17] *etc*. These studies provide a good foundation for using generative models to create scenes, but there is a general problem. They require significant expert knowledge to establish the complex definition of loss function or post-processing. CTG++ [18] provides a idea to introduce the large language model in loss function design in a user friendly way. However, this method requires repeated and complex training processes for different tasks.

In this paper, we propose DragTraffic, an instance behavior level traffic scene generator that is capable of generating realistic and diverse scenes while maintaining a high degree of freedom on controllability. To achieve realism, we employ a regress model to provide initial guesses. To account for the behavioral differences among various traffic agents, we adopt a symmetric hybrid expert architecture that imitates the real behavior of traffic participants on the road from an agent-centric perspective by using a separate model dedicated to the corresponding agent. Inspired by Draggan [19], we adopt the conditional diffusion model to achieve diversity and specific defined context, including position, velocity, heading, length, and width. All these controls can be done through dragging or typing context in an interactive and user-friendly way. Our contributions include:

- First, we propose an interactive traffic generation framework, which, to the best of our knowledge, is the first to offer a high degree of freedom for generating and editing traffic scenes.
- Second, we introduce a solution that utilizes a regression model for initial solutions, a conditional diffusion model for diversity, and a Mixture of Experts (MoE) to accommodate multiple agent types. This approach enables us to generate realistic and diverse traffic scenes with a high level of controllability.
- Third, we conduct experiments on a real-world driving dataset to evaluate the performance of DragTraffic. The results show that DragTraffic outperforms existing methods in terms of authenticity, diversity, and freedom, among other metrics.

II. Related Work

A. Trajectory Prediction

Current methods for testing and developing autonomous driving systems, such as scenario replay and rule-based approaches, have limitations in accuracy, generalization ability, and adaptive updating. These methods also require a large amount of expert knowledge to build the rules. To address these shortcomings, recent studies have explored deep learning-based motion prediction methods that can model multi-modal traffic scenes [4] [6] [20] [21]. These methods can be broadly categorized into supervised learning and generative learning approaches. Supervised learning trains a model with logged trajectories with supervised losses such as L2 loss. One of the challenges is to model inherent multi-modal behavior of the agents. For instance, Trajeglish [4] employs the template sets to help the model generate realistic multi-modal trajectories by providing a structured framework for interactions. A series works, MultiPath++ [22], DenseTNT [23] and GANet [24] use static anchors or learned goals to represent the multiple hypothises. GoHome [25] and YNet [26] predict future occupancy heatmaps, and then decode trajectories from the samples. Many of these approaches use ensembles for further diversified predictions. The next section covers generative approaches.

B. Generative models for Scene Generation

Generative models have made significant progress in sequence generation tasks such as text, pictures, and videos. Some prior works have explored the use of generative models to create traffic scenes. For example, GANs have been used to generate multiple trajectories for traffic agents in [14] and [15]. Variational Autoencoders (VAEs) have been employed in MTG [27] and CVAE-H [28] to extract representations of historical trajectories of agents and generate future trajectories. CTG [18] uses a conditional diffusion model for generating controllable traffic simulations, while CTG++ [18] introduces a scene-level conditional diffusion model guided by language instructions. MotionDiffuser [16] and SceneDM [17] both use diffusion-based models for predicting multi-agent motion and achieving state-of-the-art results on the Waymo Open Motion Dataset and Waymo Sim Agents Benchmark, respectively. While the above methods have achieved good performance, they often require professionals to design complex optimization constraints and loss functions. In contrast, our proposed framework simplifies the task of scene generation by allowing users to control the generation process through simple drag and click, while ensuring the quality of the generated scenes.

III. Problem formulation

Our aim is to generate the expected future motions for agents in a scenario. We adopt a structured vectorized representation to depict the map and agents. The trajectory of a specific agent is denoted as $\boldsymbol{\tau}_{0:t} = \{\boldsymbol{s}_0, \boldsymbol{s}_1, ..., \boldsymbol{s}_t\}$, where $\boldsymbol{s}_t \in \mathbb{R}^D$ indicating the states including the type, location, heading angle, velocity at time step t. The road map is denoted as $\boldsymbol{L} = \{\boldsymbol{l}_i\}$, where $\boldsymbol{l}_i \in \mathbb{R}^{N\times H}$ representing i_{th} lane has N segments and each segment has H lane semantic attributes (*e.g.*, intersections and crosswalks). Specifically, for the task of existing scenario augmentation or editing, we aim to generate t_F steps future trajectories

$$\boldsymbol{\tau}_{1:t_F} = f(\boldsymbol{s}_0^0 ... \boldsymbol{s}_0^M, \boldsymbol{s}_{t_F}^0, \boldsymbol{L}),$$

where ${\boldsymbol{s}_0}^1 ... {\boldsymbol{s}_0}^M$ indicates the initial states of M agents. $\boldsymbol{s}_{t_F}^0$ represents the condition information. For the task of new scenario creation, we aim to generate t_F steps future trajectories following

$$\boldsymbol{\tau}_{1:t_F} = f(G(\boldsymbol{L}), \boldsymbol{s}_{t_F}^0, \boldsymbol{L}),$$

where $G(\cdot)$ indicates the initial states generation, which can be simply achieved through dragging, typing or an agent placement module.

IV. Methodology

A. Conditional Diffusion Model Preliminaries

Diffusion models consist of a diffusion process that gradually transforms a data distribution into unstructured noise and a reverse process to recover the data distribution [29]. The forward diffusion process acting on $\boldsymbol{\tau}_{1:F}$ is defined as

$$\begin{aligned} q(\boldsymbol{\tau}_{1:F}^{1:k} | \boldsymbol{\tau}_{1:F}^0) &:= \prod_{k=1}^{k} q(\boldsymbol{\tau}_{1:F}^k | \boldsymbol{\tau}_{1:F}^{k-1}), \\ q(\boldsymbol{\tau}_{1:F}^k | \boldsymbol{\tau}_{1:F}^{k-1}) &:= \mathcal{N}(\boldsymbol{\tau}_{1:F}^k; \sqrt{1-\beta_k}\boldsymbol{\tau}_{1:F}^{k-1}, \beta_k \boldsymbol{I}), \end{aligned}$$

where the variance schedule $\beta_1, \beta_2, \cdots \beta_k$ is fixed and determines the amount of noise injected at each diffusion step, leading to a gradual corruption of the signal into an isotropic Gaussian distribution. To generate trajectories, we aim to reverse this diffusion process by utilizing a learned conditional denoising model, which is iteratively applied starting from sampled noise. Given the context information $\boldsymbol{c}(\boldsymbol{\tau}_0, \boldsymbol{s}_{t_F}^0, \boldsymbol{L})$, the reverse diffusion process is

$$\begin{aligned} p_\theta(\boldsymbol{\tau}_{1:F}^{0:k} | \boldsymbol{c}) &:= p(\boldsymbol{\tau}_{1:F}^k) \prod_{k=1}^{k} p_\theta(\boldsymbol{\tau}_{1:F}^{k-1} | \boldsymbol{\tau}_{1:F}^k, \boldsymbol{c}), \\ p_\theta(\boldsymbol{\tau}_{1:F}^{k-1} | \boldsymbol{\tau}_{1:F}^k, \boldsymbol{c}) &:= \mathcal{N}(\boldsymbol{\tau}_{1:F}^{k-1}; \boldsymbol{\mu}_\theta(\boldsymbol{\tau}^k, k, \boldsymbol{c}), \boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}_{1:F}^k, k, \boldsymbol{c})). \end{aligned}$$

The distribution $p(\boldsymbol{\tau}_{1:F}^k)$ is a normal distribution and θ denotes the parameters of the diffusion model. In this work, we adopt the idea proposed in [30] and use the conditional diffusion model as a refinement module as shown in Figure 2. This is to say, a skip connection is used to generate $\boldsymbol{\tau}_{1:F}^0$ following

$$\begin{aligned} \boldsymbol{\tau}_{1:F}^k \sim \boldsymbol{\tau}_{1:F}^\star &= f_{\text{init}}(\boldsymbol{c}), \\ \boldsymbol{\tau}_{1:F}^\gamma &= f_{\text{denoise}}(\boldsymbol{\tau}_{1:F}^{\gamma+1}, \boldsymbol{c}), \gamma = k-1, \cdots, 0, \end{aligned}$$

where $f_{\text{init}}(\cdot)$ is a standard motion forecasting model providing the initial gueses for better regression performance purpose, we will elaborate its effectiveness in following experiments section.

B. Context Description

Due to the high degree of freedom of our generation scheme, the context description can consist of some or all of the following information: vehicle type, length and width, starting position, starting speed, starting orientation, target position, target speed, target orientation.

C. Initial Trajectory Generation

We utilize Multipath++ [22] as the initial backbone for trajectory generation, which employs multi-context gating (MCG) blocks. MCG can be seen as an approximation of cross-attention. Instead of having each of the n elements attend to all m elements of the other set, MCG condenses the other set into a single context vector. The prediction heads take $\boldsymbol{c}$ then output the future $\boldsymbol{\tau}$ and K probabilities. This initial regression module is trained by minimizing the MSE loss of the predicted trajectory which is the closest to the ground truth trajectory.

D. Diffusion Refinement

Here, we detail the design of the denoising module $f_{\text{denoise}}(\cdot)$, which denoises the trajectory $\boldsymbol{\tau}_{1:F}^{\gamma+1}$ conditioned on context $\boldsymbol{c}(\boldsymbol{\tau}_0, \boldsymbol{s}_{t_F}^0, \boldsymbol{L})$. The denoising module consists of two trainable components: a MCG-based context encoder $f_{\text{context}}(\cdot)$ that learns a social-temporal embedding for the current states and condition states and a noise estimation module $f_{\boldsymbol{\epsilon}}(\cdot)$ that estimates the noise to be reduced. The γ_{th} denoising step follows

$$\boldsymbol{\epsilon}_\theta^\gamma = f_{\boldsymbol{\epsilon}}\left(\boldsymbol{\tau}_{1:F}^{\gamma+1}, f_{\text{context}}(\boldsymbol{c}), \gamma+1\right), \tag{1}$$

$$\boldsymbol{\tau}_{1:F}^\gamma = \frac{1}{\sqrt{\alpha_\gamma}}\left(\boldsymbol{\tau}_{1:F}^{\gamma+1} - \frac{1-\alpha_\gamma}{\sqrt{1-\bar{\alpha}_\gamma}}\boldsymbol{\epsilon}_\theta^\gamma\right) + \sqrt{1-\alpha_\gamma}\mathbf{z}, \tag{2}$$

where $\alpha_\gamma := 1 - \beta_\gamma$ and $\bar{\alpha}_\gamma := \prod_{i=1}^{\gamma} \alpha_i$ are parameters in the diffusion process and $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ is a noise.

E. Training Objective

To achieve the agent type sensive performance, we use a mixture of experts strucure, which includes a gate on the top to switch the suitable model according to the certain agent type. Then the scene generation framework served by symmetric models. Here for simplication purpose, we only illustrate one of them in details. To train a the dragtraffic model, we consider a two stage training strategy, where the first stage trains a denoising module and the second stage focuses on a leapfrog initializer. The reason for using two stages is to make the training more stable. The noise estimation loss is

$$\mathcal{L}_{\text{NE}} = \|\boldsymbol{\epsilon} - f_{\boldsymbol{\epsilon}}(\boldsymbol{\tau}_{1:F}^{\gamma+1}, f_{\text{context}}(\mathbf{c}), \gamma+1)\|_2,$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \boldsymbol{I})$ and the diffused trajectory $\boldsymbol{\tau}_{1:F}^{\gamma+1} = \sqrt{\bar{\alpha}_\gamma}\boldsymbol{\tau}_{1:F}^0 + \sqrt{1-\bar{\alpha}_\gamma}\boldsymbol{\epsilon}$. We backpropagate this loss to update the parameters in the context encoder $f_{\text{context}}(\cdot)$ and the noise estimation module $f_\epsilon(\cdot)$. In the second stage, we optimize the model by employing a trainable initializer model, while the denoising modules remain frozen. The loss function is

$$\mathcal{L} = \quad \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{NLL}}. \tag{3}$$

We employ a distance-based loss to minimize displacement error and use the Negative Log-Likelihood Loss function to optimize scores, both of which are commonly utilized in motion forecasting tasks.

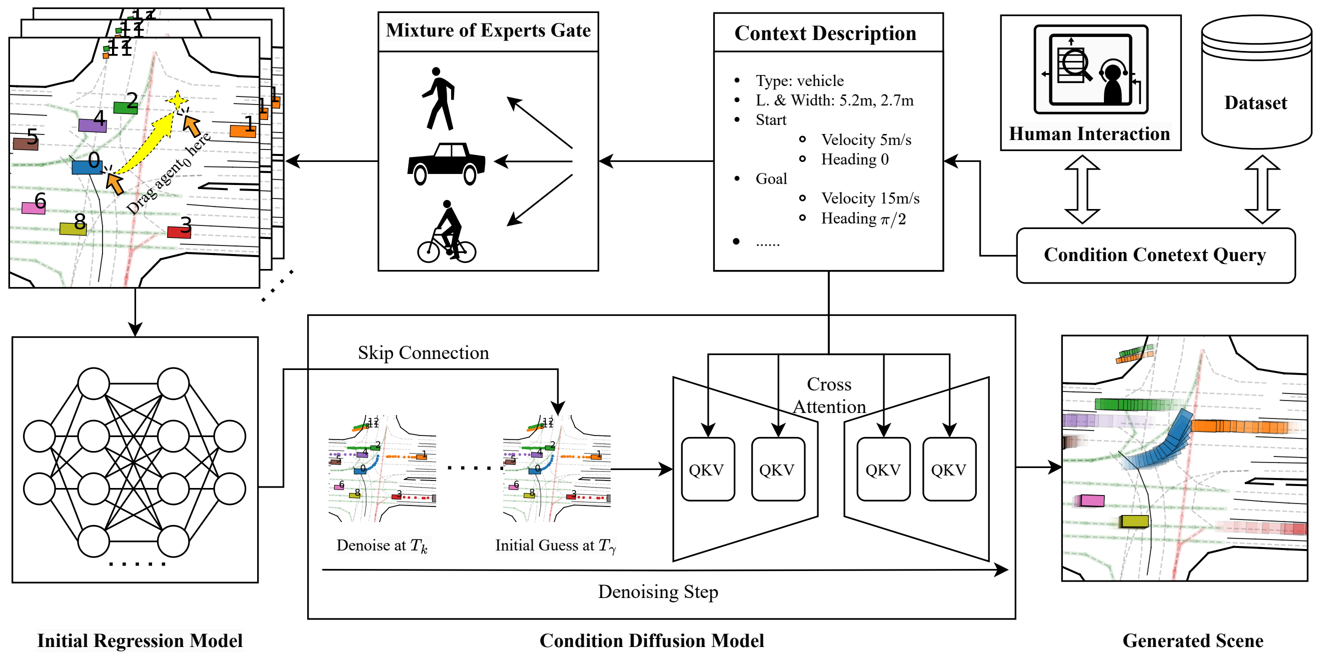


Fig. 2: **The generation pipeline.** The Condition Context Query gathers personalized information from the user, either through an interactive UI or by retrieving it from the dataset. The Mixture of Experts Gate selects the appropriate model for inference based on the agent type. Input data is presented as agent-centric vectors. After obtaining the initial solution, it is further refined through diffusion to ultimately generate the scene.

V. Experimental Results

A. Dataset

We utilize the Waymo Open Dataset [31] to train Drag-Traffic. It consists of around 70,000 scenarios, each with 20-second traffic information. To optimize the dataset for our purposes, we split each 20-second scenario into 6-second intervals and removed scenarios with less than 32 agents. We then cropped a rectangular area with a 120-meter side length centered on the ego agent and classified scenarios into three datasets: ego-centered, cyclist-centered, and pedestrian-centered. We further filtered out scenarios with less than 30 frames and invalid end points, resulting in 49,884, 29,046, and 9,344 cases, respectively. We then split the remaining cases into training, non-overlapping validation, and test datasets in an 80%, 10%, 10% ratio. To ensure fair comparison with other methods, we benchmarked the trained models on the test set and followed the placement and generation pipeline of TrafficGen, which we considered a robust baseline. Our evaluation produced both quantitative and qualitative results.

B. Metrics

To evaluate the performance of our framework, we employed two metrics: scenario collision rate (SCR) and motion forecasting related metrics. The SCR measures the consistency of the generated vehicle's behaviors by calculating the average percentage of vehicles that collide with others in each scenario. We consider two vehicles as colliding if their bounding boxes overlap above a predefined IOU threshold. For the open-loop evaluation, we used the common metrics in trajectory prediction tasks: MinADE$_k$, MinFDE$_k$, Heading error and Speed error. These metrics are calculated based on the trajectory with the cloest endpoint to the ground truth over k predictions. Since our condition context includes endpoint information, we treated the point before the last one as the endpoint when calculating the MinFDE.

C. Implementation details

During the model training phase, we pre-train the conditional diffusion model for 100 epochs and then freeze its parameters. Similar to [30], after obtaining the initial guess from the regression model, we employ the standard U-net diffusion model to carry out 5 steps of denoising. The total number of denoising steps is 100 for both training and testing. For the Initializer model, we add an MLP layer on top of the backbone to encode the condition information of dimension 8 and obtain hidden features of length 1024, which are then concatenated with the outputs of the state encoder and the lane encoder. We fine-tune the diffusion model and the Initializer model together for 40 epochs. All other baselines that do not utilize the diffusion model are trained for 150 epochs, with the one showing the lowest loss selected for experimental evaluation. All models are trained with a learning rate decay starting at 0.0003 and decaying to 0.0004, using 4 RTX 3090 GPUs.

D. Results and discussion

We believe that an outstanding scene generator should possess two essential characteristics. Firstly, it must generate realistic and reasonable trajectories by providing accurate context. Similar to the trajectory prediction task, we use

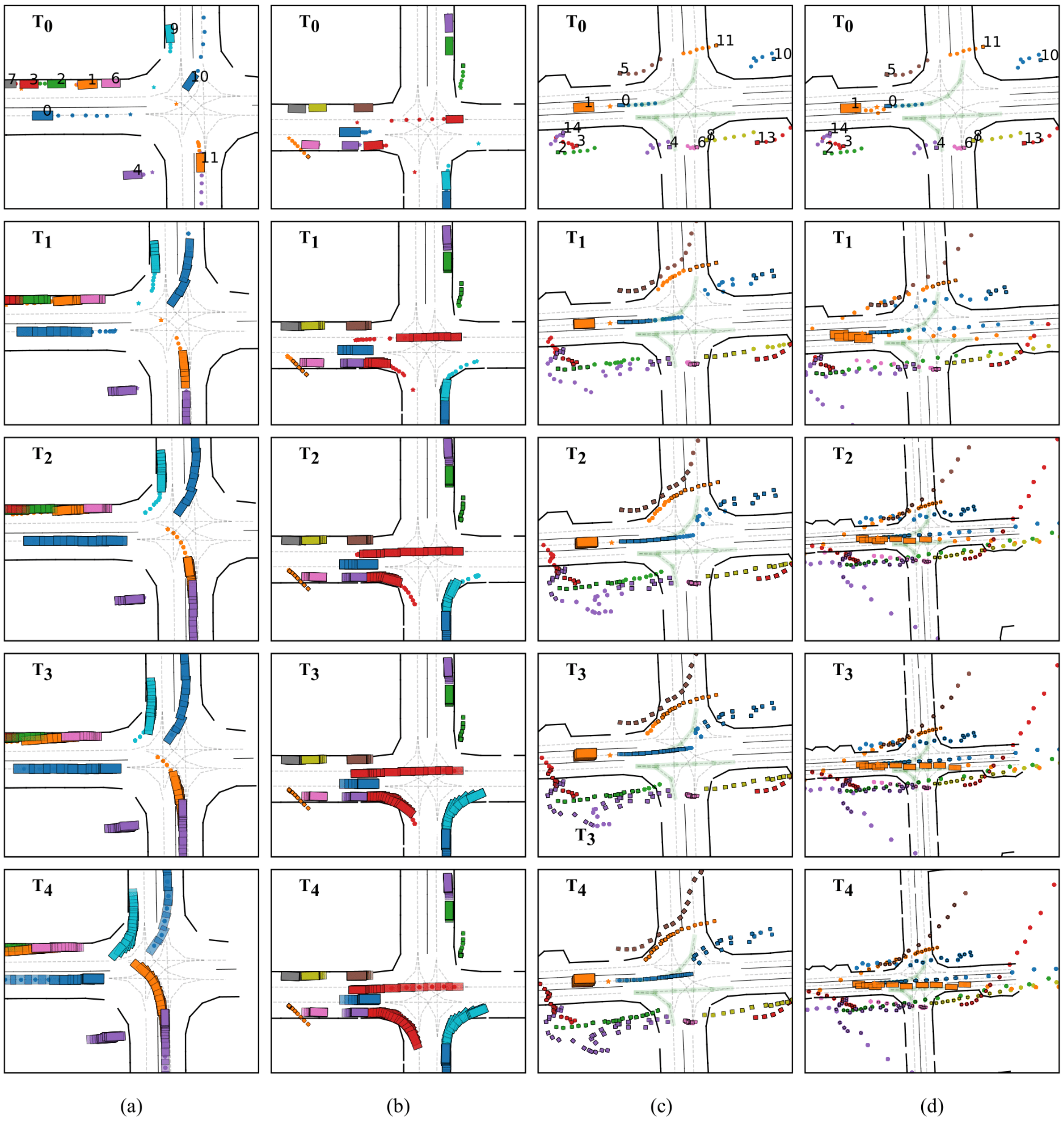


Fig. 3: **The demonstration of creating, editing and correction.** Colored boxes represent agents, with different sizes for each type. A motorcycle is depicted as a long bar, while a pedestrian is represented as a square. A series of colored dots in front of each agent indicates the generated trajectory, and the shadow represents past actions.

displacement error metrics to evaluate its performance. Secondly, a remarkable scene generator should offer a high degree of freedom and controllability, enabling users to create realistic scenes while exploring rare and high-value corner cases. This is where DragTraffic excels compared to other existing frameworks. In the following sections, we will elaborate on the experimental results, focusing particularly on these two characteristics.

1) Scene Generation: To assess the diversity of scenario generation, we conducted three subsets of tests on MinADE and MinFDE for different agent types. The results, presented in the Table I, demonstrate that our approach achieves good predictive performance for various agents and can approximate the ground truth. In contrast, plain TrafficGen performs well for vehicle agents but exhibits significant performance degradation for pedestrians and bicycles due

TABLE I: Scene Generation Quality Evaluation

Dataset	Model	Min ADE$_6$	Min FDE$_6$	Heading Error	Speed Error
Vehicle	TrafficGen	3.32	5.41	0.05	0.05
	TrafficGen Mixture Training with Condition	3.09	4.40	**0.04**	0.05
	DragTraffic with Condition	**2.53**	**3.33**	0.05	**0.03**
Pedestrian	TrafficGen	6.50	9.04	0.10	0.40
	TrafficGen Mixture Training with Condition	2.05	3.53	0.02	**0.36**
	DragTraffic with Condition	**1.59**	**2.74**	**0.02**	0.40
Cyclist	TrafficGen	17.05	23.62	0.41	0.31
	TrafficGen Mixture Training with Condition	**3.34**	**4.31**	**0.06**	**0.21**
	DragTraffic with Condition	3.93	6.21	0.07	0.37

to its reliance on vehicle data, which oversimplifies the problem and reduces model generalizability. To ensure fairness, we trained trafficGen on a mixed dataset, which also yielded promising results on some metrics. This highlights the importance of establishing a framework for different agent types. DragTraffic outperforms other models on both vehicle and pedestrian datasets, thanks to its MoE structure, which enhances its generalizability. However, we observe that DragTraffic does not perform well on the Cyclist dataset, possibly because the proportion of cyclists in the scene data is too small. In Table II, we present the results of the Interaction Reasoning Evaluation with sampling intervals of 3s, 6s, and 9s. We collected 300 scenes with high interactivity between different agents for evaluation, including 200 scenes related to pedestrians and 100 scenes related to cyclists. The performance of DragTraffic under the three sampling conditions is superior to the baseline. Interestingly, we observe that the SCR varies with the sampling interval, which is contrary to the results obtained in TrafficGen [8]. However, we believe this is reasonable because multiple rollouts can introduce more cumulative errors.

2) Scene Editing & Inpaiting: Fig. 3 demonstrates the quality of scenes generated by DragTraffic based on existing data. We use the information of current frame and the 90th frame in the dataset as conditions. The left-turn and right-turn scenes at the intersection where the most interactions occur are respectively shown in columns (a) and (b). These scenes reflect courtesy and competition for right of way among different agents. For example, in (a), Agent 9 and Agent 11 engage in a fierce competition for the right of way, while Agent 0 gives way. This shows that DragTraffic can simulate traffic participants in different situations under highly dynamic and complex traffic intersections to make reasonable, smooth, and realistic future actions, reflecting real-world characteristics. Next, we verify that this capability is not limited to simple log replay in more complex scenarios (c) and (d). We focus on agent 1, which is waiting at an intersection for the motorcycle in front to start. However, the movement of a large number of pedestrians around interferes with the decision-making of agent 1, which has a conservative driving style. As a result, it continues to watch the movements of pedestrians even when the motorcycle is already driving to the intersection, leading to the phenomenon of robot freezing, which is common in the fields of autonomous driving and robotics. To address this issue, we design similar scenarios by setting the condition information for agent 1 (100 meters in front of it, longitudinal speed of 20m/s, and latitudinal speed of -2m/s) to encourage more proactive behavior. As shown in (d), the results generated by DragTraffic demonstrate effective control over scene editing. Note that we only show the control of agent 1 for the convenience of explanation, but in fact, DragTraffic allows us to control the generation of multiple agents simultaneously. Unlike other generation control methods that rely on complex optimization constraints and loss designs, our generation processes require only simple dragging or typing interactions, showcasing the superiority of this framework.

TABLE II: Interaction Reasoning Evaluation

Rollout Interval	Model	SCR(%)
3s	T.G. Mixture Training with Condition	15.00
	DragTraffic with Condition	9.20
6s	T.G. Mixture Training with Condition	14.43
	DragTraffic with Condition	8.97
9s	T.G. Mixture Training with Condition	13.59
	DragTraffic with Condition	3.33

VI. Future Work

While our proposed framework can generate realistic and diverse traffic scenarios, there are still areas for improvement. For example, we can implement post-processing or sampling techniques to ensure that the generated driving actions adhere to dynamic constraints. Another promising avenue for future work is to develop a multi-round generation process incorporating human feedback.

VII. CONCLUSIONS

In this paper, we introduced DragTraffic, a generalized, interactive, and controllable traffic scene generation framework based on conditional diffusion. Our framework addresses the limitations of existing scene generation methods in terms of controllability, accuracy, and versatility, enabling non-experts to create a variety of realistic driving scenarios for different types of traffic agents. We achieve this using an adaptive mixture expert architecture with a regression model for initial solutions, followed by refinement through the conditional diffusion model to ensure diversity. The denoising process incorporates user-customized context via cross-attention, enhancing controllability. The qualitative and quantitative results on a real-world driving dataset demonstrate the effectiveness of DragTraffic, which can significantly aid in the evaluation and training of autonomous driving systems by providing diverse and scalable corner cases.

REFERENCES

[1] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[3] 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," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[4] J. Philion, X. B. Peng, and S. Fidler, "Trajeglish: Learning the language of driving scenarios," *arXiv preprint arXiv:2312.04535*, 2023.

[5] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-centric trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[6] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[7] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun, "Scenegen: Learning to generate realistic traffic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 892–901.

[8] L. Feng, Q. Li, Z. Peng, S. Tan, and B. Zhou, "Trafficgen: Learning to generate diverse and realistic traffic scenarios," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3567–3575.

[9] H. Zhang, H. Song, S. Li, M. Zhou, and D. Song, "A survey of controllable text generation using transformer-based pre-trained language models," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–37, 2023.

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[11] A. R. et al, "Hierarchical text-conditional image generation with clip latents," 2022.

[12] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," 2022.

[13] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[14] W. Ding, B. Chen, B. Li, K. J. Eun, and D. Zhao, "Multimodal safety-critical scenarios generation for decision-making algorithms evaluation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1551–1558, 2021.

[15] Z.-H. Yin, L. Sun, L. Sun, M. Tomizuka, and W. Zhan, "Diverse critical interaction generation for planning and planner evaluation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7036–7043.

[16] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, D. Anguelov, *et al.*, "Motiondiffuser: Controllable multi-agent motion prediction using diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9644–9653.

[17] Z. Guo, X. Gao, J. Zhou, X. Cai, and B. Shi, "Scenedm: Scene-level multi-agent trajectory generation with consistent diffusion models," *arXiv preprint arXiv:2311.15736*, 2023.

[18] Z. Zhong, D. Rempe, Y. Chen, B. Ivanovic, Y. Cao, D. Xu, M. Pavone, and B. Ray, "Language-guided traffic simulation via scene-level diffusion," in *Conference on Robot Learning*. PMLR, 2023, pp. 144–177.

[19] X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, and C. Theobalt, "Drag your gan: Interactive point-based manipulation on the generative image manifold," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–11.

[20] S. Wang, Y. Chen, J. Cheng, X. Mei, R. Xin, Y. Song, and M. Liu, "Improving autonomous driving safety with pop: A framework for accurate partially observed trajectory predictions," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 450–14 456.

[21] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7577–7586.

[22] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. Lam, D. Anguelov, and B. Sapp, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," *CoRR*, vol. abs/2111.14973, 2021. [Online]. Available: https://arxiv.org/abs/2111.14973

[23] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," *CoRR*, vol. abs/2108.09640, 2021. [Online]. Available: https://arxiv.org/abs/2108.09640

[24] M. Wang, X. Zhu, C. Yu, W. Li, Y. Ma, R. Jin, X. Ren, D. Ren, M. Wang, and W. Yang, "Ganet: Goal area network for motion forecasting," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1609–1615.

[25] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "GOHOME: graph-oriented heatmap output for future motion estimation," *CoRR*, vol. abs/2109.01827, 2021. [Online]. Available: https://arxiv.org/abs/2109.01827

[26] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," *CoRR*, vol. abs/2012.01526, 2020. [Online]. Available: https://arxiv.org/abs/2012.01526

[27] W. Ding, W. Wang, and D. Zhao, "Multi-vehicle trajectories generation for vehicle-to-vehicle encounters," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[28] G. Oh and H. Peng, "Cvae-h: Conditionalizing variational autoencoders via hypernetworks and trajectory forecasting for autonomous driving," *arXiv preprint arXiv:2201.09874*, 2022.

[29] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[30] W. Mao, C. Xu, Q. Zhu, S. Chen, and Y. Wang, "Leapfrog diffusion model for stochastic trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5517–5526.

[31] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.