

Diverse Controllable Diffusion Policy With Signal Temporal Logic

Yue Meng[✉], *Graduate Student Member, IEEE*, and Chuchu Fan[✉], *Member, IEEE*

Abstract—Generating realistic simulations is critical for autonomous system applications such as self-driving and human-robot interactions. However, driving simulators nowadays still have difficulty in generating controllable, diverse, and rule-compliant behaviors for road participants: Rule-based models cannot produce diverse behaviors and require careful tuning, whereas learning-based methods imitate the policy from data but are not designed to follow the rules explicitly. Besides, the real-world datasets are by nature “single-outcome”, making the learning method hard to generate diverse behaviors. In this letter, we leverage Signal Temporal Logic (STL) and Diffusion Models to learn controllable, diverse, and rule-aware policy. We first calibrate the STL on the real-world data, then generate diverse synthetic data using trajectory optimization, and finally learn the rectified diffusion policy on the augmented dataset. We test on the NuScenes dataset and our approach can achieve the most diverse rule-compliant trajectories compared to other baselines, with a runtime 1/17X to the second-best approach. In the closed-loop testing, our approach reaches the highest diversity, rule satisfaction rate, and the least collision rate. Our method can generate varied characteristics conditional on different STL parameters in testing. A case study on human-robot encounter scenarios shows our approach can generate diverse and closed-to-oracle trajectories.

Index Terms—Autonomous agents, autonomous vehicle navigation, machine learning for robot control.

I. INTRODUCTION

REALISTIC behavior modeling is vital for developing simulators and studying intelligent systems such as autonomous driving and warehouse ground robots [1], [2]. To close the sim-to-real gap for the agents, it is critical to model the uncertainty and rule adherence properties that naturally arise from human behaviors. For example, human drivers have different characteristics (aggressiveness, conservativeness), which affect their decision-making in challenging scenarios (e.g., going through a roundabout with dense traffic). Besides, low-level

driving commands (steering the wheel, accelerating, braking) are also driven by high-level maneuvers (lane-keeping, lane-changing) and traffic rules (e.g., speed limit). Thus, it is of paramount importance to endow agent models with diversity, controllability, and rule-awareness.

However, driving simulators up-to-date [3], [4] still struggle in delivering diverse and rule-compliance agent behaviors. They either use recorded trajectories or utilize rule-based or imitation-based methods to generate policy. Rule-based approaches (IDM [5], MOBIL [6]) directly encode rules into mathematical models thus can provide safety and goal-reaching performance. Still, they assume simplified driving scenarios and require careful parameter tuning, lacking diversity and real-tickness. Imitation-based methods [7], [8] learn from real-world driving data, being more akin to human behaviors, but are prone to violate the rules. Besides, since there is only one outcome (out of many possible future trajectories) per scene in the ground truth, only a limited diversity is achieved by these imitation-based approaches [9].

Impeding the advancement of learning realistic behaviors are three challenges: (1) A flexible rule representation, (2) the scarcity of multiple-outcome datasets, and (3) the trade-off between rule compliance and diversity. Our letter systematically addresses these problems by leveraging a formal language termed Signal Temporal Logic (STL) [10], [11]. STL is known for modeling complicated rules [12], and there are increasing works recently studying controller synthesis under STL specifications via trajectory optimization [13], deep learning [14], [15] and reinforcement learning [16]. Inspired by these works and recent breakthroughs in diffusion models [17] for policy learning [18], we proposed a parametric-STL approach to flexibly encode traffic rules, augment the dataset, and learn a controllable diffusion policy to balance quality and diversity¹.

The whole pipeline is: We first specify rules via parameter-STL and use demonstrations to calibrate the parameters. The parameters involve both discrete and continuous values, adding the capacity to form multi-modal and diverse policy distributions. Based on the STL, the parameters, and the original data, we generate the “multiple-outcome” data via trajectory optimization. Next, we use Denoising Diffusion Probabilistic Model (DDPM [17]) to learn from the augmented data. Finally, different from other diffusion-based policies [18], [19], an additional neural network is designed to regulate the trajectories to be rule-compliant and diverse.

¹“Diversity” refers to generate **different** trajectories for the **same** STL rule.

Manuscript received 26 March 2024; accepted 23 July 2024. Date of publication 16 August 2024; date of current version 23 August 2024. This article was recommended for publication by Associate Editor Rudolf Lioutikov and Editor Jens Kober upon evaluation of the reviewers’ comments. This work was supported in part by the National Science Foundation (NSF) under Grant #CCF-2238030 and in part by the MIT-Ford Alliance Program. (*Corresponding author: Yue Meng.*)

The authors are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: mengyue@mit.edu; chuchu@mit.edu).

The annotation tool, augmented dataset, and code are available at <https://github.com/mengyue/pSTL-diffusion-policy>.

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2024.3444668>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3444668

We conduct experiments on NuScenes [20], a large-scale autonomous driving dataset. We first label the dataset using our annotation tool and generate the augmented dataset. Then we train our approach on the augmented dataset and evaluate on the validation set and in closed-loop testing. Our approach results in the highest STL satisfaction rate on the validation set and generates the most diverse trajectories compared to baselines. In closed-loop testing, our approach reaches the highest overall performance regarding diversity, STL satisfaction, collision, out-of-lane, and progress. We also show that with varied STL parameters, our approach can reflect different driver characteristics in a challenging roundabout scenario, which is valuable for diverse behavior modeling in simulators. A case study on human-robot encounter scenarios also shows similar performance compared to other baselines.

To summarize, our contributions are: (1) We are the first to use a parametric-STL formulation to augment the driving dataset for diverse and controllable policy generation (2) we propose an add-on module (RefineNet) for Diffusion Models to improve trajectory diversity and quality (3) we achieve leading performance in the open-loop evaluation and closed-loop test on NuScenes [20] (4) our algorithm, annotation tool and the augmented data will be available via open-source distribution.

II. RELATED WORK

Trajectory prediction: Decades of effort have been devoted to exploring trajectory prediction for autonomous systems [21]. Traditional methods include physics-based methods [22] and machine-learning approaches such as Gaussian Process [23]. More recent works use neural networks to conduct behavior cloning or imitation learning [24] on multi-modal large-scale datasets (NuScenes [20], WOMB [25]), where the performance is improved by better scene representations (rasterization [7], polylines [26]), advanced architectures [27], and varied output types (sets [28], heatmaps [29] and distributions [30]). Most works predict trajectories by fitting the dataset. Instead, we assume the rules are given, extract statistics from the data and learn diverse and rule-compliant trajectories conditional on these statistics.

Diverse trajectory generation: Deep generative models (Variational Auto Encoder, Generative Adversarial Network, and Diffusion Models) are used to learn to produce diverse trajectories [18], [19], [27], [31], [32]. The diversity is either learned implicitly from data [18], [19], [31], [32] or guided by the “Minimum over N” (MoN) loss [27]. To avoid the prediction concentrating merely around the major mode of the data, the work in [33] learns diversity sampling functions (DSF) in the latent space to generate diverse trajectories. In the autonomous driving domain, recent works use inductive heuristics from the scenes (such as drivable area [34] or lanes [35]) to further regulate the diverse behaviors to be “reasonable” in common sense. However, it is worth noting that the original dataset lacks diversity in essence (for each scene, there is just one ground trajectory). Unlike all methods that learn diversity from the original data, we first generate diverse data using trajectory optimization then learn the diverse policy. The closest works

similar to ours are ForkingPath [9] and [36], where the former is for pedestrian prediction and requires heavy annotation, and the latter generates data using IDM which are less diverse.

Realistic agent modeling: Realisticness is often achieved by augmenting imitation with common sense factors, such as collision-free [2], [37], map-consistency [38], attractor-repeller effect [30], driving patterns [39], and LLM-based designs [40]. Recently, Signal Temporal Logic (STL) [10] is widely used to specify rules for trajectory predictions [18], [41], for its expressiveness to encode rules [41] and differentiable policy learning [14], [15]. We follow this line of work and parametrize the STL to learn controllable behaviors.

III. PRELIMINARIES

A. Signal Temporal Logic (STL)

A signal $s = x_t, x_{t+1}, \dots, x_{t+T}$ is a discrete-time finite sequence of states $x_i \in \mathbb{R}^n$. STL is a formal language to specify signal properties via the following expressions [11]:

$$\phi ::= \top \mid \mu(x) \geq 0 \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathbf{U}_{[a,b]} \phi_2 \quad (1)$$

where the boolean-type (“true” / “false”) operators split by “|” serve as building blocks to construct an STL formula. Here \top is “true”, μ denotes a function $\mathbb{R}^n \rightarrow \mathbb{R}$, and $\neg, \wedge, \mathbf{U}, [a, b]$ are “not”, “and”, “until”, and time interval from a to b . Other operators are “or”: $\phi_1 \vee \phi_2 = \neg(\neg\phi_1 \wedge \neg\phi_2)$, “imply”: $\phi_1 \Rightarrow \phi_2 = \neg\phi_1 \vee \phi_2$, “eventually”: $\Diamond_{[a,b]}\phi = \top \mathbf{U}_{[a,b]}\phi$ and “always”: $\Box_{[a,b]}\phi = \neg\Diamond_{[a,b]}\neg\phi$. Denote $s, t \models \phi$ if a signal s from time t satisfies ϕ , i.e., ϕ returns “true”. It is easy to check satisfaction for $\top, \mu \geq 0, \neg, \wedge$, and \vee . As for temporal operators [42]: $s, t \models \Diamond_{[a,b]}\phi \Leftrightarrow \exists t' \in [t+a, t+b] \ s, t' \models \phi$ and $s, t \models \Box_{[a,b]}\phi \Leftrightarrow \forall t' \in [t+a, t+b] \ s, t' \models \phi$.

Robustness score $\rho(s, t, \phi)$ measures how well a signal s satisfies ϕ , where $\rho \geq 0$ if and only if $s, t \models \phi$. A larger ρ reflects a greater satisfaction margin. The calculation is [10]:

$$\begin{aligned} \rho(s, t, \top) &= 1, \quad \rho(s, t, \mu) = \mu(s(t)) \\ \rho(s, t, \neg\phi) &= -\rho(s, t, \phi) \\ \rho(s, t, \phi_1 \wedge \phi_2) &= \min\{\rho(s, t, \phi_1), \rho(s, t, \phi_2)\} \\ \rho(s, t, \Diamond_{[a,b]}\phi) &= \sup_{r \in [a,b]} \rho(s, t+r, \phi) \\ \rho(s, t, \Box_{[a,b]}\phi) &= \inf_{r \in [a,b]} \rho(s, t+r, \phi) \end{aligned} \quad (2)$$

In our letter, we adopt a differentiable approximation for ρ proposed in [43] to provide gradient-based policy guidance.

B. Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Models are powerful generative models that learn a density distribution from the training data to generate samples that resemble these data. A diffusion model consists of two procedures: forward process (diffusion) and inverse process (denoising). In letter [17], during the diffusion process, the data are iteratively fused with Gaussian noise until they are close to the white noise. A neural network is trained to predict the noises added to these samples at different diffusion steps. To generate

samples, in the denoising process, the latent samples initialized from the white noise are recovered iteratively by “removing” the noise predicted by the network.

IV. TECHNICAL APPROACH

A. Problem Formulation

Consider an autonomous system, where we denote the state of the agent at time t as $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$, the control $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$, the scene context $c \in \mathcal{C} \subseteq \mathbb{R}^p$, and the STL template $\Psi : \Gamma \times \mathcal{C} \rightarrow \Phi$ which generates the STL formula $\phi \in \Phi$ based on the STL parameters $\gamma \in \Gamma$ and the scene provided (i.e., $\phi = \Psi(\gamma, c)$). Assume a known differentiable discrete-time system dynamics: $f : \mathcal{S} \times \mathcal{U} \rightarrow \mathcal{S}$, where from s_0 we could generate a trajectory $\tau = (s_0, s_1, \dots, s_T) \in \mathbb{R}^{(T+1) \times n}$ based on control sequence $(u_0, u_1, \dots, u_{T-1}) \in \mathcal{U}^T$. Assume a known diversity measure $J : \Xi \rightarrow \mathbb{R}$ which is a real-valued function over a set of trajectories Ξ .

Given a set of demonstrations $\mathcal{D} = \{(c^i, \tau^i)\}_{i=1}^N$ and an STL template Ψ , the goal is first to generate a set of STL parameters $\{\gamma_i\}_{i=1}^N$ such that $\tau_i, 0 \models \Psi(\gamma_i, c_i), \forall i = 1, 2, \dots, N$, and secondly, learn a policy $\pi : \mathcal{S} \times \mathcal{C} \times \Gamma \rightarrow \mathcal{U}^T$ such that for a given initial state s_0 , a scene context state c and STL parameters γ , the trajectories set Ξ generated by $\pi(s_0, c, \gamma)$ can both (1) satisfy the STL rule $\tau, 0 \models \Psi(\gamma, c), \forall \tau \in \Xi$, and (2) maximize the diversity measure $J(\Xi)$ (we will use entropy to measure the diversity and the detailed computation for the entropy is shown in Section V-B).

B. System Modeling and STL Rules for Autonomous Driving

Ego car model: We use a unicycle model for the dynamics: $x_{t+1} = x_t + v_t \cos(\theta_t) \Delta t$, $y_{t+1} = y_t + v_t \sin(\theta_t) \Delta t$, $\theta_{t+1} = \theta_t + w_t \Delta t$, $v_{t+1} = v_t + a_t \Delta t$ where the state $s_t = (x_t, y_t, \theta_t, v_t)^T$ stands for ego car's xy coordinates, heading angle, and the velocity at time t , and the controls $u_t = (w_t, a_t)^T$ are angular velocity and acceleration. We denote the ego car width W and length L .

Scene model: The scene context consists of neighbor vehicles and lanes. We consider the N_n nearest neighbors within the R -radius disk centered at the ego vehicle and denote their states at time t as $\mathcal{N} = \{I_t^j, x_t^j, y_t^j, \theta_t^j, v_t^j, L^j, W^j\}_{j=1}^{N_n}$, where the binary indicator I_t^j is one when the j^{th} neighbor is valid and zero otherwise (this happens when there are less than N_n neighbor vehicles within the R -radius disk), and the rests are similar to the ego car state. We represent each lane as $Q = (I, x_1, y_1, \theta_1, \dots, x_{N_p}, y_{N_p}, \theta_{N_p}) \in \mathbb{R}^{3N_p+1}$ with an indicator to show its validity and then a sequence of N_p centerline waypoints with 2D coordinates and directions. We denote the current lane, the left adjacent lane, and the right adjacent lane for the ego vehicle as Q_c , Q_l , and Q_r .

STL rules and parameters: We define common STL rules that are required in autonomous driving scenarios,

$$\phi_0 = \square_{[0,T]} \mathbf{v}_{\min} \leq \text{Speed}(s) \leq \mathbf{v}_{\max}$$

$$\phi_1 = \square_{[0,T]} \text{Dist}(s, \mathcal{N}) \geq d_{\text{safe}}$$

$$\phi_2 = \square_{[0,T]} d_{\min} \leq \text{Dist}(s, Q_c) \leq d_{\max}$$

$$\phi_3 = \square_{[0,T]} |\text{Angle}(s, Q_c)| \leq \theta_{\max}$$

$$\phi_4^H = \diamond_{[0,T]} \square_{[0,T]} d_{\min} \leq \text{Dist}(s, Q_H) \leq d_{\max}$$

$$\phi_5^H = \diamond_{[0,T]} \square_{[0,T]} |\text{Angle}(s, Q_H)| \leq \theta_{\max} \quad (3)$$

where the variables in blue are STL parameters, ϕ_0 is for the speed limit, ϕ_1 specifies the distance to the neighbors should always be greater than the threshold. Formulas ϕ_2 and ϕ_3 restrict the vehicle's distance and heading deviation from the current lane, whereas ϕ_4^H and ϕ_5^H are for the left and right adjacent lanes with $H \in \{l, r\}$ to restrict the car to eventually keep the distance/angle deviation from the target lane within the limit². We consider the driving mode \mathcal{M} belongs to one of the high-level behaviors: lane-keeping ($\mathcal{M} = 0$), left-lane-change ($\mathcal{M} = 1$) and right-lane-change ($\mathcal{M} = 2$). Denote the STL parameters $\gamma = (\mathcal{M}, v_{\min}, v_{\max}, d_{\text{safe}}, d_{\min}, d_{\max}, \theta_{\max})^T \in \Gamma \subseteq \mathbb{R}^7$. The STL template thus is defined as:

$$\begin{aligned} \Psi(\gamma) = & ((\mathcal{M} = 0) \Rightarrow (\phi_0 \wedge \phi_1 \wedge \phi_2 \wedge \phi_3)) \\ & \wedge ((\mathcal{M} = 1) \Rightarrow (\phi_0 \wedge \phi_1 \wedge \phi_4^l \wedge \phi_5^l)) \\ & \wedge ((\mathcal{M} = 2) \Rightarrow (\phi_0 \wedge \phi_1 \wedge \phi_4^r \wedge \phi_5^r)) \end{aligned} \quad (4)$$

C. STL Parameters Calibration

Given the STL template Ψ and the expert demonstrations \mathcal{D} , we find STL parameters γ_i for each trajectory τ_i and the scene c_i such that $\tau_i, 0 \models \Psi(\gamma_i, c_i)$. We develop an annotation tool to manually label the high-level behaviors \mathcal{M} for all the trajectories, then based on \mathcal{M} , we calibrate the rest of the STL parameters by making the robustness score on the expert trajectories equal to zero. This is done by conducting min/max extraction on the existing measurements: e.g., to find d_{\min} , we first check the high-level policy, then compute the minimum distance from the trajectory τ to the target lane.

D. Diverse Data Augmentation

After obtaining the STL parameters for each scene, we augment the original demonstration by generating diverse behaviors using trajectory optimization. For each scene c_i and parameter γ_i , we formulate the following optimization:

$$\begin{aligned} & \underset{u_0, u_1, \dots, u_{T-1}}{\text{minimize}} \quad \sigma_+(-\rho((s_0, s_1, \dots, s_T), 0, \Psi(\gamma_i, c_i))) \\ & \text{subject to} \quad u_{\min} \preceq u_t \preceq u_{\max}, \forall t = 0, 1, \dots, T-1 \\ & \quad \quad \quad s_{t+1} = f(s_t, u_t), \forall t = 0, 1, \dots, T-1 \end{aligned} \quad (5)$$

where $\sigma_+(\cdot) = \max(\cdot, 0)$, $a \preceq b$ means the vector a is element-wise no larger than the vector b , and u_{\min}, u_{\max} are predefined control limits. The system dynamics and STL formula make (5) a nonlinear optimization, and we use a gradient-based method to solve for the solution. To increase solutions' diversity, we consider all three driving modes for \mathcal{M} , and for each mode, we run gradient-descent from K initial solutions uniformly sampled from the solution space \mathcal{U}^T . We denote our augmented dataset as

²Since we aim to eventually “always” keep it in the limit, the “Always” operator is put inside the “Eventually” scope to realize this behavior.

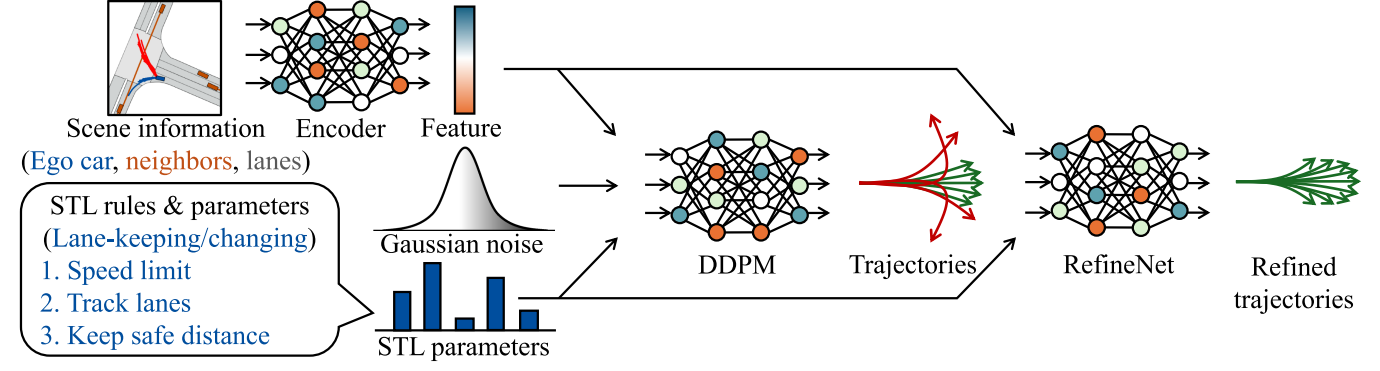


Fig. 1. Learning framework. The neural encoder embeds the scene to a feature vector. The DDPM takes the feature vector, the STL parameters (indicating driving modes, speed limit, safe distance threshold, etc) and the Gaussian noise to generate trajectories. RefineNet takes the upstream trajectories and features and generates diverse and rule-compliant trajectories.

$\tilde{D} = \{(\tilde{c}_i, \tilde{\gamma}_i, \{\tilde{\tau}_i^j\}_{j=1}^K)\}_{i=1}^{3N}$ where $\tilde{c}_i = c_{\lfloor i/3 \rfloor}$ and $\tilde{\gamma}_i = \gamma_{\lfloor i/3 \rfloor}$. Here, $\lfloor \cdot \rfloor$ denotes rounding a float number to an integer index.

E. Policy Learning Framework

Given the new dataset \tilde{D} , we learn the diverse and rule-compliant behavior via the learning framework shown in Fig. 1. The encoder network embeds the ego state and the scene to a feature vector. The DDPM network takes the feature vector, STL parameters, and a Gaussian noise to produce trajectories that closely match the distribution in \tilde{D} . Finally, the RefineNet takes upstream features and trajectories to generate diverse and rule-compliant trajectories.

Encoder Network: The ego state and the scene are first transformed to the ego frame, \tilde{s} and $\tilde{c} = \{\tilde{Q}_c, \tilde{Q}_l, \tilde{Q}_r, \tilde{N}\}$ accordingly. The state \tilde{s} is sent to a fully connected network (FCN) to get ego feature: $z_{ego} = g_{ego}(\tilde{s}) \in \mathbb{R}^d$. Similarly, the lanes are fed to a lane FCN to generate feature: $z_{lane} = [g_{lane}(\tilde{Q}_c), g_{lane}(\tilde{Q}_l), g_{lane}(\tilde{Q}_r)] \in \mathbb{R}^{3d}$, where $[\cdot, \dots]$ is the vector concatenation. To make the neighbors feature not depend on the neighbor orders, we utilize permutation-invariant operators in [44] with a neighbor FCN g_{nei} to get: $z_{nei} = [\max_j g_{nei}(\tilde{N}_j), \min_j g_{nei}(\tilde{N}_j), \sum_j g_{nei}(\tilde{N}_j)] \in \mathbb{R}^{3d}$, where \tilde{N}_j is the j^{th} neighbor feature. The final merged embedding is: $z = [z_{ego}, z_{lane}, z_{nei}] \in \mathbb{R}^{7d}$.

DDPM Network: Given the embedding z , the STL parameters γ , the sample $\tau^{(0)}$ from \tilde{D} , the random noise ϵ , we generate the diffused samples: $\tau^{(t)} = \sqrt{\alpha_t}\tau^{(0)} + \sqrt{1 - \alpha_t}\epsilon$ for uniformly sampled diffusion steps $t \sim \text{Uniform}(1, T_d)$ and pre-defined coefficients α_t and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. The DDPM network $g_d : \mathbb{R}^{7d+7+2T+1} \rightarrow \mathbb{R}^{T \times 2}$ takes $z, \gamma, \tau^{(t)}, t$ as input and predicts the noise, guided by the diffusion loss in the first stage of the training:

$$\mathcal{L}_d = \mathbb{E}_{\tilde{D}, t} \left[\left\| \epsilon - g_d(z, \gamma, \tau^{(t)}, t) \right\|^2 \right] \quad (6)$$

In inference, from the Gaussian noise $\tau^{(T_d)} \sim \mathcal{N}(0, I)$, the trajectories are generated iteratively by the denoising step: $\tau^{(t-1)} = \frac{1}{\sqrt{\alpha_t}}(\tau^{(t)} - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}g_d(z, \gamma, \tau^{(t)}, t)) + \sigma_t \xi_t$ with $\xi_t \sim$

$\mathcal{N}(0, I)$ and $\sigma_1 = 0$ and $\sigma_t = 1$ for $t \geq 2$. We denote the trajectories generated by DDPM as τ_d .

Refine Network: After DDPM is trained, we use RefineNet, a fully-connected network $g_r : \mathbb{R}^{7d+7+2T} \rightarrow \mathbb{R}^{T \times 2}$, to regulate the trajectories generated by the DDPM network to encourage rule-compliance and diversity. RefineNet takes as input the trajectories with the highest STL score from the last five denoising steps and outputs a residual control sequence conditional on the violation of the STL rules, which is:

$$\tau_{final} = \tau_d + \mathbb{1}\{\rho(\tau_d, 0, \Psi(\gamma, c)) < 0\} \cdot g_r(z, \gamma, \tau_d) \quad (7)$$

If the DDPM produced trajectories already satisfy the STL rules, the RefineNet will not affect the final trajectories (i.e., $\tau_{final} = \tau_d$); otherwise, the RefineNet improves trajectories' diversity and the rule satisfaction rate.³ It is hard to directly optimize for the diversity measure (entropy approximation requires state space discretization, which is non-differentiable). Instead, in the second stage, the RefineNet is updated by the following loss:

$$\mathcal{L}_r = \mathbb{E}_{\tilde{D}} \left[\text{tr} \left(I - (\mathcal{K}(\{\tau_{final, j}\}_{j=1}^{N_d}) + I)^{-1} \right) \right] \quad (8)$$

where $\text{tr}(\cdot)$ is the matrix trace, and $\mathcal{K} \in \mathbb{R}^{N_d \times N_d}$ is the Direct Point Process (DPP) kernel [33] over N_d samples:

$$\mathcal{K}_{ij} = \mathbb{1}(\rho(\tau_i) \geq 0) \cdot \exp(-|\tau_i - \tau_j|^2) \cdot \mathbb{1}(\rho(\tau_j) \geq 0). \quad (9)$$

Minimizing (8) increases the trajectory cardinality and quality [33], thus increases the diversity and rule satisfaction.

F. Guidance-Based Online Policy Refinement

In evaluation, our learned policy might violate the STL rules in the unseen scenarios due to the generalization error. Similar to [18], we use the STL guidance to improve the sampling process. For a new state s_0 , scene c , the embedding z and parameter γ in testing, we replace the original denoising step to $\tau^{(t-1)} = \frac{1}{\sqrt{\alpha_t}}\tilde{\tau}^{(t)} + \sigma_t \xi_t$, where $\tilde{\tau}^{(t)}$ is initialized as $\tau^{(t)} - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}g_d(z, \gamma, \tau^{(t)}, t)$ and is updated by minimizing $-\rho(\tilde{\tau}^{(t)}, 0, \Psi(\gamma, c))$ via gradient descent. The work [18]

³Table I shows that DDPM results in a very low rule satisfaction rate, which reflects the great potential of using RefineNet for improvement.

conducts multiple guidance steps at every denoising step. In contrast, we find it sufficient to just conduct the guidance step at the last several denoising steps in the diffusion model to accelerate the computation. Although nonlinear optimization does not guarantee optimality, in practice, this method can satisfy the rules with high probability, as shown below.

V. EXPERIMENTS

We first conduct experiments on NuScenes where our method generates the most diverse trajectories quantitatively and visually compared to baselines, reaching the highest STL compliance rate. In closed-loop test we get the lowest collision rate and out-of-lane rate. Visualization shows how varied STL parameters affect the agent behavior under the same scene, indicating our approach's potential for diverse agent modeling. We also consider a human-robot scenario, where we generate the most close-to-oracle distribution.

A. Implementation Details

Dataset: NuScenes [20] is a large-scale real-world driving dataset that comprises 5.5 hours of driving data from 1000 scenes (from Boston and Singapore). We use the “trainval” split of the dataset (850 scenes), densely sample from all valid time instants and randomly split the dataset with 70% for training (11763 samples) and 30% for validation (5042 samples). We developed an annotation tool and it took a student four days to label high-level driving behaviors for the data. We mainly focus on vehicles and leave the other road participants (pedestrians and cyclists) for future research.

Algorithm details: The planning horizon is $T = 20$, the duration is $\Delta t = 0.5$ s, the number of neighbors is $N_n = 8$, perception radius is $R = 50$ m, each lane has $N_p = 15$ waypoints, and the control limits are $u_{\max} = -u_{\min} = (0.5 \text{ rad/s}, 5.0 \text{ m/s}^2)^T$. In data generation, the number of samples per scene is $K = 64$. Similar to [18], the diffusion steps is $T_d = 100$ and a cosine variance schedule is used. The networks are FCN with 2 hidden layers, with 256 units for each layer and a ReLU activation for the intermediate layers. For our method, we first generate the augmented dataset, then train the DDPM for 500 epochs using (6). Finally, we freeze the Encoder and DDPM and train the RefineNet for 500 epochs using (8). We use PyTorch with an ADAM [45] optimizer, a learning rate 3×10^{-4} and a batch size 128. The augmentation takes 5 hours, and training takes 8 hours on an RTX4090Ti GPU.

B. Open-Loop Evaluation

Baselines: The methods are **Traj. Opt.**: Trajectory optimization solution (treated as “Oracle”); **VAE**: Variational Auto-encoder; **DDPM**: Trained with the loss in (6); **TrafficSim** [2]: train a VAE with an extra rule-violation loss; and **CTG** [18]: train the DDPM and test with guidance during sampling. And **Ours**: our method (Section IV-E); **Ours+guidance**: with guidance (Section IV-F). For ablations, **Ours (w/o RefineNet)**: no RefineNet; and **Ours (\mathcal{L}_{STL})**: uses $\mathcal{L}_{STL} = \text{ReLU}(0.5 - \rho)$ to train the RefineNet. We implement baselines to accommodate

for modality and STL rules, and also train VAE and DDPM on the original NuScenes to show gains from our augmentation.

Metrics: We evaluate the trajectories by (1) **Success**: the ratio of the scenes that have at least one trajectory satisfying the STL rules, (2) **Compliance**: the ratio of generated trajectories satisfying the STL rules (valid trajectories), (3) **Valid area**: The 2d occupancy area of the valid trajectories averaged over all the scenes, (4) **Entropy**: At each time step, we compute the entropy for the normalized angular velocity and the acceleration from the valid trajectories respectively, and average over all time steps and all scenes, and (5) **Time**: measures the trajectory generation time.

Quantitative results: As shown in Table I, both VAE and DDPM trained on our augmented dataset achieve a higher diversity (valid area and entropy) than them trained on the original NuScenes data, indicating the value of our augmentation technique to generate diverse demonstrations. VAE and DDPM's low compliance rates (less than 10%) imply the need to use an advanced model. Compared to advanced baselines TrafficSim [2] and CTG [18], “Ours” strikes a sharp rise in the quality and diversity: 32 – 66% higher for rule compliance, 36 – 198% larger valid area, and up to 33% increase in entropy. Moreover, “Ours+guidance” achieves the highest quality and diversity with 1/17X the time used by the best baseline CTG⁴.

Ablation studies: “Ours (w/o RefineNet)” is a bit better than “DDPM”, where the gains result from the ensemble of DDPM outputs from the last five denoising steps. With RefineNet and the STL loss used, “Ours (\mathcal{L}_{STL})” gets a 47 – 188% increase in the diversity measure compared to “Ours (w/o RefineNet)”. Further using the diversity loss, “Ours” achieves a 19 – 22% increase in diversity measure, and “Ours+guidance” generates the most diverse trajectories but at the cost of 4X longer inference time. We can see that adding the RefineNet and using a diversity loss greatly improves the diversity and rule compliance rate (though using the loss \mathcal{L}_r will drop the compliance rate by 3%.)

Visualizations: In Fig. 2 we plot all the rule-compliant trajectories (generated by different methods) under specific scenes and color them based on high-level driving modes (red for “right-lane-change”, blue for “lane-keeping” and green for “left-lane-change”). “Ours” and “Ours+guidance” generate close to *Traj. Opt.* distributions, with the largest area coverage among all learning baselines.

C. Closed-Loop Testing

Implementation: We select 26 challenging trials in NuScenes dataset (where the ego car needs to avoid cars on the street or to keep track of curvy lanes), and we set the STL parameters to the minimum/maximum values in the training data to represent the largest feasible range for parameter selection. We start the simulation from these trials and stop it if (1) it reaches the max simulation length or (2) collision happens or the ego car drives out-of-lane. Due to the fast-changing environment, we follow the MPC [15] rather than windowed-policy [19], [32] or other

⁴“Ours+guidance” is much faster than CTG because we only use guidance at the last five denoising steps, whereas CTG uses guidance at every step.

TABLE I

OPEN-LOOP EVALUATION: THE HIGHEST IS SHOWN IN **BOLD** AND THE SECOND BEST IS SHOWN IN UNDERLINE. OUR DATA AUGMENTATION BOOSTS THE DIVERSITY FOR BASELINES VAE AND DDPM. “OURS+GUIDANCE” GENERATES THE TRAJECTORIES IN THE HIGHEST QUALITY (SUCCESS AND COMPLIANCE) AND DIVERSITY (VALID AREA, ENTROPY), WITH THE RUNTIME 1/17X TO THE SECOND BEST CTG [18]

Methods	Augmentation	Success \uparrow	Compliance \uparrow	Valid area \uparrow	Entropy \uparrow	Time (s) \downarrow
<i>Traj. Opt.</i>	Yes	0.961	0.746	49.130	2.124	36.205
VAE	-	0.337	0.077	0.618	0.162	0.036
VAE	Yes	0.253	0.018	0.627	0.200	<u>0.039</u>
DDPM [17]	-	0.514	0.078	1.760	0.455	0.081
DDPM [17]	Yes	0.548	0.050	3.444	0.557	0.081
TrafficSim [2]	Yes	0.699	0.335	6.798	1.059	0.037
CTG [18]	Yes	<u>0.833</u>	0.267	14.933	1.384	13.582
Ours (w/o RefineNet)	Yes	0.624	0.078	5.899	0.780	0.172
Ours (\mathcal{L}_{STL})	Yes	0.817	0.573	17.032	1.152	0.174
Ours	Yes	0.782	0.442	<u>20.284</u>	<u>1.411</u>	0.174
Ours+guidance	Yes	0.840	<u>0.544</u>	33.530	1.735	0.786

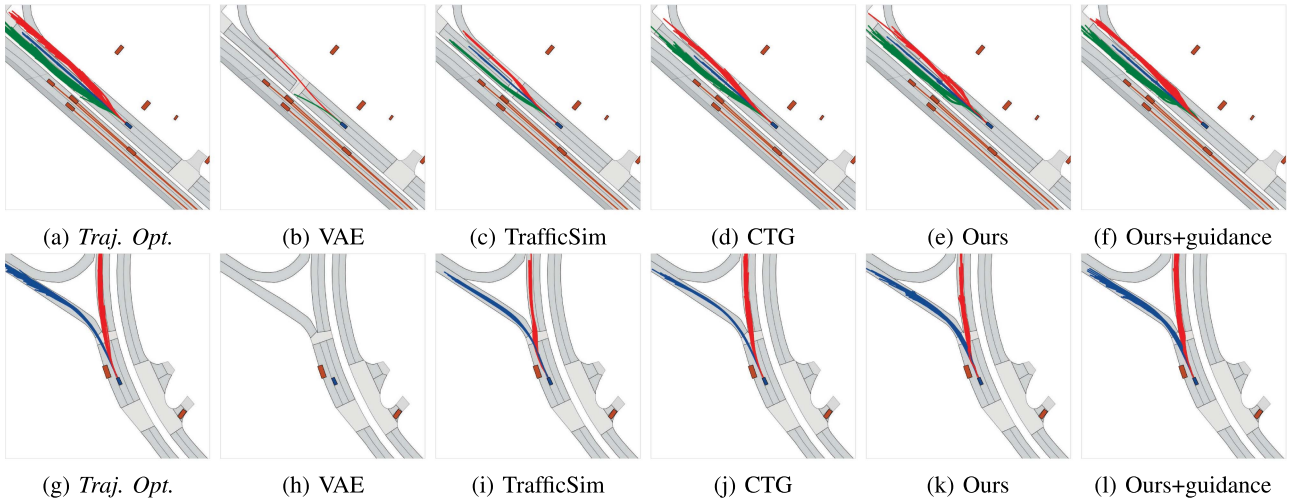


Fig. 2. Open-loop visualizations (Green: “left-lane-change”, red: “right-lane-change” and blue: “lane-keeping”). Our approach generates the closest to the *Traj. Opt.* solution and results in the largest trajectory coverage among all the learning methods.

mechanism⁵ [31]. At every time step, out of the 64 generated trajectories, we choose the one with the highest robustness score and pick its first action to interact with the simulator. We measure: (1) **Compliance**: the ratio of generated trajectories satisfying the STL rules, (2) **Valid Area**: The 2d occupancy area of the valid trajectories, (3) **Progress**: the average driving distance of the ego car, (4) **Collision**: the ratio of trials ending in collisions, (5) **Out-of-lane**: the ratio of trials ending in driving out-of-lane, and (6) **Time**: the runtime at every step.

Results: As shown in Table II, our approach without guidance already achieves high performances compared to VAE, DDPM,

⁵The works [19], [31], [32] are mainly for robot manipulation, where the task horizon is long and the environment is relatively static. One challenge in driving scenarios is that the environment can change suddenly in planning (a new neighbor vehicle emerges, lane changes, etc). Thus, the windowed policy might not react to these changes and a mechanism to detect the change and trigger the replanning process is needed. We do not use the goal-conditioned mechanism in [31] as STL cannot be fully conveyed by a few goal states.

and TrafficSim [2], with slightly high computation time compared to these learning-based baselines - the overhead in the runtime mainly owes to the DDPM and STL evaluation. Given that the simulation $\Delta t = 0.5$ s, this overhead is still in a reasonable range. With the guidance used, “Ours+guidance” surpasses all the baselines in quality and diversity metrics (except for CTG’s progress), achieving the lowest collision rate and zero out-of-lane. Compared to the best baseline, CTG, our runtime is just 1/24X of CTG’s. This shows our method’s ability to provide diverse and high-quality trajectories with an acceptable time budget in tests.

Diverse behaviors under different STL parameters: To show the controllability, we use our method with varied STL parameters to render agent behaviors in a challenging scene shown in Fig. 3, where the ego car waits to join in a roundabout with dense traffic on the right side. We assign three different maximum speed limits to our network, fix the minimum speed to 0 m/s, and plot the scenario at $t = 22$. In Fig. 3, the history of the ego car

TABLE II

CLOSED-LOOP TESTING: THE HIGHEST IS SHOWN IN **BOLD** AND THE SECOND BEST IS SHOWN IN UNDERLINE. “OURS+GUIDANCE” STRIKES IN DIVERSITY AND RULE COMPLIANCE WITH THE LEAST COLLISION AND OUT-OF-LANE RATE WITHIN AN ACCEPTABLE COMPUTATION BUDGET

Methods	Compliance \uparrow	Valid area \uparrow	Progress \uparrow	Collision \downarrow	Out-of-lane \downarrow	Time (s) \downarrow
VAE	0.076	1.403	71.435	0.385	0.154	<u>0.019</u>
DDPM	0.168	5.250	83.493	<u>0.115</u>	0.000	0.031
TrafficSim [2]	0.311	3.363	71.113	0.269	0.077	0.018
CTG [18]	<u>0.704</u>	<u>16.838</u>	92.181	<u>0.115</u>	0.000	9.280
Ours	0.448	12.908	74.189	0.192	0.154	0.055
Ours+guidance	0.763	21.577	<u>88.638</u>	0.077	0.000	0.379

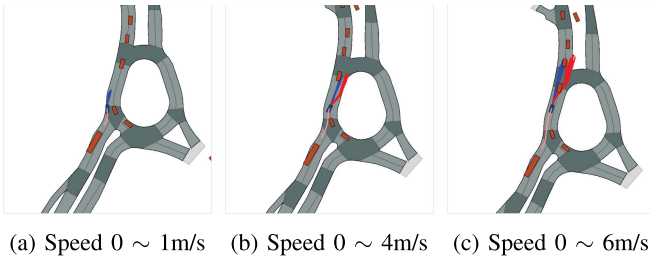


Fig. 3. Diverse behaviors due to varied STL parameters. When the speed limit is low, the agent waits until all vehicles pass the roundabout. When at the middle-speed limit, the agent joins the queue in the middle but yields to other vehicles at high speed. At the high-speed limit, the car joins the queue and keeps its place as traversing the roundabout.

is in pink, and planned trajectories are in blue (for lane-keeping) and red (for right-lane-change). When the max speed limit is low (1 m/s), the agent waits until all cars finish the roundabout. When at a speed range [0 m/s, 4 m/s], the agent joins the queue in the middle but yields to other vehicles at high speed. When at a wider interval ([0 m/s, 6 m/s]), the car joins the middle of the queue and keeps the place as moving in the roundabout. This finding shows we can model different agent characteristics, which is valuable for realistic agent modeling in simulators.

D. Case Study on Diverse Human Behaviors Generation

We demonstrate how to generate diverse human behaviors in scenarios [46] where the STL rule for the human is to reach the goal while avoiding collision with the incoming robot. We choose collision thresholds as STL parameters and calibrate them on 1000 real-world trajectories collected by [46]. The dataset is further augmented as in Section IV-C. We train all methods on the augmented dataset for comparison.

In Table III, “Ours+guidance” is the highest in “Area” and “Entropy”, reaching the most close-to-oracle (*Traj. Opt.*) performance with 1/35X of its time. Although TrafficSim [2] reaches the highest compliance rate, its trajectories are less diverse (shown in Fig. 4(c)). “Ours+guidance” gets 24 ~ 38% improvement over CTG [18] in rule-compliance rate and diversity, being 7.7X faster in inference speed. Visualizations from Fig. 4 show that the trajectory distribution of Ours_G (Ours+guidance) is close to the oracle. These results show our advantage in generating diverse rule-compliant policy.

TABLE III

OPEN-LOOP EVALUATION IN HUMAN-ROBOT ENCOUNTER SCENARIOS [46]. “OURS+GUIDANCE” REACHES THE CLOSEST DIVERSITY TO THE ORACLE (*TRAJ. OPT.*), WHILE USING ONLY 1/35X OF ITS TIME

Methods	Compliance \uparrow	Area \uparrow	Entropy \uparrow	Time (s) \downarrow
<i>Traj. Opt.</i>	<i>1.000</i>	<i>6.114</i>	<i>55.553</i>	<i>66.125</i>
VAE	0.420	1.894	18.600	0.002
DDPM [17]	0.262	4.673	20.715	0.023
TrafficSim [2]	0.966	2.076	17.998	<u>0.002</u>
CTG [18]	0.665	4.645	39.492	16.153
Ours	0.811	<u>5.675</u>	<u>44.715</u>	0.040
Ours+guidance	<u>0.921</u>	5.773	48.959	1.850

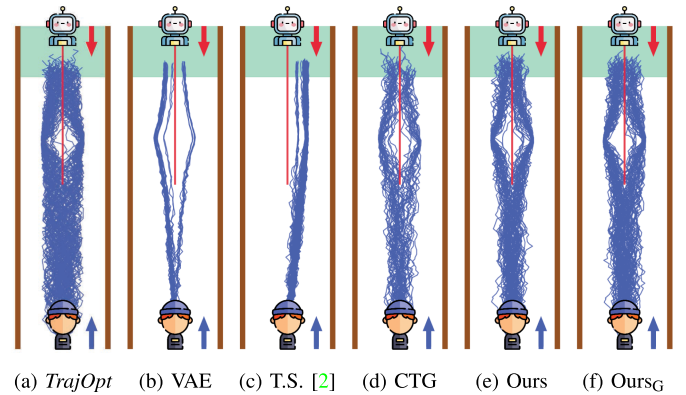


Fig. 4. Valid trajectories for the human-robot encounters. VAE and T.S. (TrafficSim) cannot capture diverse trajectories, whereas Ours_G (Ours+guidance) are close-to-oracle.

VI. CONCLUSION

We propose a method to learn diverse and rule-compliant agent behavior via data augmentation and Diffusion Models. We model the rules as Signal Temporal Logic (STL), calibrate the STL parameters from the dataset, augment the data using trajectory optimization, and learn the diverse behavior via DDPM and RefineNet. In the NuScenes dataset, we produce the most diverse and rule-compliant trajectories, with 1/17X the runtime used by the second-best baseline [18]. In closed-loop test, we achieve the highest safety and diversity, and with varied STL parameters we can generate distinct agent behaviors. A

case study on human-robot scenarios shows we can generate closed-to-oracle trajectories. The limitations are: high-level behavior labeling effort; longer runtime than VAE, DDPM, or TrafficSim [2]; and lack of guarantees. Besides, in rare cases if DDPM learns rule-compliance and non-diverse trajectories, our RefineNet cannot improve the diversity. We plan to address those and more complex rules in the future.

ACKNOWLEDGMENT

Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and don't reflect the views of the sponsors.

REFERENCES

- [1] W. Jager and M. Janssen, "The need for and development of behaviourally realistic agents," in *Proc. Int. Workshop Multi-Agent Syst. Agent-Based Simul.*, Springer, 2002, pp. 36–49.
- [2] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "TrafficSim: Learning to simulate realistic multi-agent behaviors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10400–10409.
- [3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, PMLR, 2017, pp. 1–16.
- [4] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. IEEE 21st Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2575–2582.
- [5] A. Kesting and M. T. D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philos. Trans. Roy. Soc. A*, vol. 368, pp. 4585–4605, 2010.
- [6] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transp. Res. Rec.*, vol. 1999, no. 1, pp. 86–94, 2007.
- [7] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 683–700.
- [8] R. Bhattacharyya et al., "Modeling human driving behavior through generative adversarial imitation learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 2874–2887, Mar. 2023.
- [9] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann, "The garden of forking paths: Towards multi-future trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10508–10518.
- [10] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. Formal Model. Anal. Timed Syst.*, 2010, Art. no. 92.
- [11] A. Donzé, T. Ferrere, and O. Maler, "Efficient robust monitoring for STL," in *Proc. Comput. Aided Verification: 25th Int. Conf.*, 2013, pp. 264–279.
- [12] Y. E. Sahin, R. Quirynen, and S. Di Cairano, "Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming," in *Proc. IEEE Amer. Control Conf.*, 2020, pp. 454–459.
- [13] C. Dawson and C. Fan, "Robust counterexample-guided optimization for planning from differentiable temporal logic," in *2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 7205–7212.
- [14] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," *Int. J. Robot. Res.*, vol. 42, no. 6, pp. 356–370, 2023.
- [15] Y. Meng and C. Fan, "Signal temporal logic neural predictive control," *IEEE Robot. Automat. Lett.*, vol. 8, no. 11, pp. 7719–7726, Nov. 2023.
- [16] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3834–3839.
- [17] J. Ho, A. Jain, and P. Abbeel, "Denosing diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.
- [18] Z. Zhong et al., "Guided conditional diffusion for controllable traffic simulation," in *2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3560–3566.
- [19] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," *Robot.: Sci. Syst. Conf.*, 2023.
- [20] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11621–11631.
- [21] F. Leon and M. Gavrilescu, "A review of tracking and trajectory prediction methods for autonomous driving," *Mathematics*, vol. 9, no. 6, 2021, Art. no. 660.
- [22] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *2009 IEEE 5th Int. Conf. Intell. Comput. Commun. Process.*, 2009, pp. 417–422.
- [23] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A Bayesian non-parametric approach to modeling motion patterns," *Auton. Robots*, vol. 31, pp. 383–400, 2011.
- [24] A. O. Ly and M. Akhloufi, "Learning to drive by imitation: An overview of deep behavior cloning methods," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 195–209, Jun. 2021.
- [25] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2446–2454.
- [26] J. Gao et al., "VectorNet: Encoding HD maps and agent dynamics from vectorized representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11525–11533.
- [27] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2255–2264.
- [28] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CoverNet: Multimodal behavior prediction using trajectory sets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14074–14083.
- [29] T. Gilles, S. Sabatini, D. Tsishkov, B. Stanculescu, and F. Moutarde, "Thomas: Trajectory heatmap output with learned multi-agent sampling," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [30] C. Jiang et al., "MotionDiffuser: Controllable multi-agent motion prediction using diffusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 9644–9653.
- [31] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *Robot.: Sci. Syst. Conf.*, 2024.
- [32] P. M. Scheikl et al., "Movement primitive diffusion: Learning gentle robotic manipulation of deformable objects," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 5338–5345, Jun. 2024.
- [33] Y. Yuan and K. Kitani, "Diverse trajectory forecasting with determinantal point processes," in *Proc. Int. Conf. Learn. Representations*, 2020.
- [34] Y. Xu, H. Cheng, and M. Sester, "Controllable diverse sampling for diffusion based motion behavior forecasting," in *IEEE Intell. Vehicles Symp.*, 2024.
- [35] S. Kim, H. Jeon, J. W. Choi, and D. Kum, "Diverse multiple trajectory prediction using a two-stage prediction network trained with lane loss," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 2038–2045, Apr. 2023.
- [36] M. Stoll, M. Mazzola, M. Dolgov, J. Mathes, and N. Möser, "Scaling planning for automated driving using simplistic synthetic data," 2023, *arXiv:2305.18942*.
- [37] Y. Meng, Z. Qin, and C. Fan, "Reactive and safe road user simulations using neural barrier certificates," in *2021 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6299–6306.
- [38] S. Casas, C. Gulino, S. Suo, and R. Urtasun, "The importance of prior knowledge in precise multimodal prediction," in *2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2295–2302.
- [39] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. Driggs-Campbell, and M. J. Kochenderfer, "Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning," in *2019 IEEE Int. Conf. Robot. Automat.*, 2019, pp. 789–795.
- [40] Z. Zhong et al., "Language-guided traffic simulation via scene-level diffusion," in *Proc. Conf. Robot Learn.*, 2023, pp. 144–177.
- [41] S. Maierhofer, P. Moosbrugger, and M. Althoff, "Formalization of intersection traffic rules in temporal logic," in *2022 IEEE Intell. Veh. Symp. (IV)*, 2022, pp. 1135–1144.
- [42] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Formal Techn., Modelling Anal. Timed Fault-Tolerant Syst.*, 2004, pp. 152–166.
- [43] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *2017 IEEE Conf. Control Technol. Appl.*, 2017, pp. 1235–1240.
- [44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3 d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
- [45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
- [46] A. Linard, I. Torre, A. Steen, I. Leite, and J. Tumova, "Formalizing trajectories in human-robot encounters via probabilistic STL inference," in *2021 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 9857–9862.