LTLDoG: Satisfying Temporally-Extended Symbolic Constraints for Safe Diffusion-based Planning

Zeyu Feng[†1], Hao Luan[†1], Pranav Goyal[1], and Harold Soh[1,2]

***Abstract*—Operating effectively in complex environments while complying with specified constraints is crucial for the safe and successful deployment of robots that interact with and operate around people. In this work, we focus on generating long-horizon trajectories that adhere to static and temporally-extended constraints/instructions at test time. We propose a data-driven diffusion-based framework, LTLDoG, that modifies the inference steps of the reverse process given an instruction specified using finite linear temporal logic (LTL_f). LTLDoG leverages a satisfaction value function on LTL_f and guides the sampling steps using its gradient field. This value function can also be trained to generalize to new instructions not observed during training, enabling flexible test-time adaptability. Experiments in robot navigation and manipulation illustrate that the method is able to generate trajectories that satisfy formulae that specify obstacle avoidance and visitation sequences. Code and supplementary material are available online at https://github.com/clear-nus/ltldog.**

***Index Terms*—Robot Safety, Machine Learning for Robot Control, Imitation Learning, Hybrid Logical/Dynamical Planning and Verification**

I. Introduction

Recent methodologies [1]–[3] utilizing data-driven diffusion models [4]–[6] have shown remarkable performance in generating robot behaviors across a wide range of tasks. Thanks to their ability to model complex distributions, these methods have surpassed several leading offline reinforcement learning techniques and classical model-based trajectory optimization methods, especially in long-horizon decision-making tasks [1], [3]. However, while conventional diffusion models excel at learning from training datasets, they lack the ability to adapt to new objectives or comply with new constraints during deployment. This shortcoming can lead to unsafe behaviors, posing risks to humans, robots, and their surrounding environment.

Manuscript received: May, 6, 2024; Accepted July, 22, 2024. This article was recommended for publication by Associate Editor E. Bıyık and Editor A. Faust upon evaluation of the reviewers' comments.

This research is supported by A*STAR under its National Robotics Programme (NRP) (Award M23NBK0053). The authors would also like to acknowledge partial support from a Google South Asia & Southeast Asia Award and from the National Research Foundation, Singapore under its Medium Sized Center for Advanced Robotics Technology Innovation.

[†]Z.F. and H.L. contributed equally to this work.

[1]Department of Computer Science, School of Computing, National University of Singapore, Singapore. {zeyu, haoluan, pgoyal, harold}@comp.nus.edu.sg

[2]Smart Systems Institute, National University of Singapore.

Digital Object Identifier (DOI): 10.1109/LRA.2024.3443501.

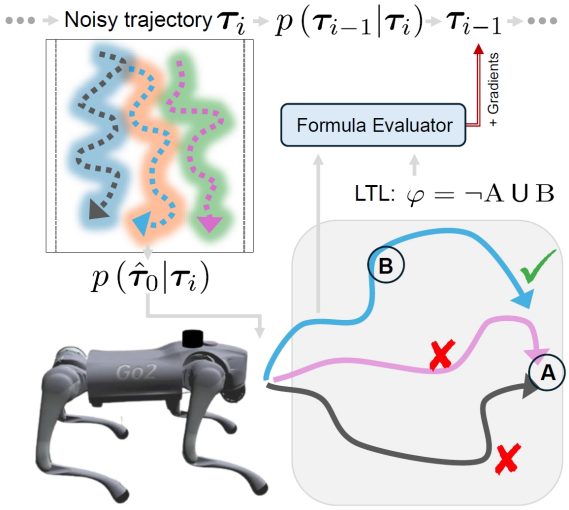


Fig. 1. We present LTLDoG, a diffusion-based planning framework for generating trajectories that comply with specified LTL_f formulae. In the example above, a robot dog is tasked to arrive at the goal position (A), but first has to visit B and avoid obstacles (crosses).

In view of this limitation, there has been very recent work on diffusing *safe* trajectories. Xiao *et al.* [7] integrated a dynamics model into the denoising diffusion process and incorporated a class of Control Barrier Functions (CBF) to meet safety criteria. Botteghi *et al.* [8] approached the issue by embedding both safety and reward considerations into a constrained optimization framework, employing CBF constraints as labels for classifier guidance. However, these approaches primarily address *static* environmental constraints. For example, while they can maneuver around obstacles on a *local* scale, they fail to comply with more complex temporally-extended directives such as "*avoid the kitchen until you are clean*".

In this work, we propose an alternative approach to flexible trajectory planning with diffusion models that is designed to satisfy both static safety requirements and temporal constraints. The core idea is to plan with diffusion models to satisfy finite linear temporal logic (LTL_f) formulae [9]. LTL_f offers the ability to define a broad spectrum of instructions/constraints that might emerge during deployment. For example, LTL_f can describe a visitation order of different objects and locations. The use of propositional logic operators, such as *not*, facilitates the delineation of safe regions within the state space.

We develop LTLDoG (**LTL D**iffusion-**o**rienting **G**uidance, pronounced "Little Dog"), a posterior-sampling based diffusion framework that accommodates finite LTL formulae at test time. We present two variants of LTLDoG: our main method,

LTLDoG-S, can be applied to the generation of finite-length trajectory in robot tasks where the labeling function for propositional events is differentiable. LTLDoG-S employs a differentiable formula checker in conditional sampling — specifically, we modify the reverse process to condition upon the criteria that the final (predicted) trajectory satisfies a given LTL_f formula. For when a differentiable labeling function is unavailable, we propose LTLDoG-R, which uses a trained LTL_f neural-symbolic evaluator for posterior sampling. Notably, both variants do not require collecting expert demonstrations for every potential LTL_f instruction. They retain the temporal compositionality and local consistency properties associated with diffusion models [1] — as long as the dataset contains a diverse set of paths, they can *potentially* "stitch together" snippets of trajectories from the training data to generate plans for similar, but unseen, LTL_f formulae.

Experiments on two benchmark environments (long-horizon planning for navigation and policy learning for manipulation) demonstrate that LTLDoG is able to generate trajectories that satisfy feasible safety and temporal constraints. We find that our methods possess the ability to re-plan alternative paths at a high-level based on given instruction. Moreover, real robot experiments show that the generated trajectories can be successfully transferred to a quadruped robot. In summary, this paper makes three key contributions:

- A conditional trajectory sampling approach designed for LTL_f instructions that leverages pre-trained diffusion models;
- A regressor-guidance neural network for diffusion that generalizes to novel LTL_f formulae within a given template structure;
- Experimental results on benchmark problems and real world demonstrations that validate the effectiveness of planning with safety and temporal constraints.

From a broader perspective, LTLDoG is the first method that fuses symbolic model checking (using LTL_f) with expressive diffusion-based generative models. We hope our results lays the groundwork towards performant, yet safer and more trustworthy robots.

II. Preliminaries and Notation

In this work, our focus is to extend diffusion-based planning methods towards generating trajectories that comply with specified LTL_f formulae. Here, we provide a concise introduction to diffusion methods in the context of planning and finite linear temporal logic.

A. Planning with Diffusion

Many tasks in planning, reinforcement learning, and imitation learning require generating trajectories under some specific objective. Let $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space, respectively. We use $\boldsymbol{\tau} = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)$ to refer to a trajectory, where T is the planning horizon. The environment transitions to a new state $\mathbf{s}_{t+1}$ when an agent executes action $\mathbf{a}_t$ at state $\mathbf{s}_t$. Let the abbreviation $\mathcal{J}(\boldsymbol{\tau}|g)$ denote the objective value function conditioned on a goal state where the trajectory must terminate at, for example, (discounted) cumulative rewards in reinforcement learning, cumulative error of actions in imitation learning, or cost for safety constraints.

Diffusion-based planning methods directly generate partial or entire trajectories by using diffusion models pre-trained on a dataset of trajectories. Let $p_0\left(\boldsymbol{\tau}^0\right)$ denote the distribution of trajectories in dataset, where $\boldsymbol{\tau}^0$ represents a noiseless trajectory. Given an N-step discrete approximation of forward diffusion process $p\left(\boldsymbol{\tau}^i|\boldsymbol{\tau}^{i-1}\right)$ that slowly corrupts data by adding prespecified noise, diffusion models learn an iterative denoising procedure by approximating the score function $\nabla_{\boldsymbol{\tau}^i} \log p_i\left(\boldsymbol{\tau}^i\right)$ using a step-dependent neural network s_θ trained with denoising score matching [10]:

$$\theta^* = \arg\min_\theta \mathbb{E}_{i,\boldsymbol{\tau}^i,\boldsymbol{\tau}^0}\left[\left\|s_\theta\left(\boldsymbol{\tau}^i, i\right) - \nabla_{\boldsymbol{\tau}^i} \log p\left(\boldsymbol{\tau}^i|\boldsymbol{\tau}^0\right)\right\|^2\right], \tag{1}$$

in which $i \sim \mathcal{U}\{1, 2, \ldots, N\}$ is the diffusion timestep, and $\boldsymbol{\tau}^i \sim p\left(\boldsymbol{\tau}^i|\boldsymbol{\tau}^0\right)$ is the trajectory $\boldsymbol{\tau}^0$ corrupted with noise. Throughout the paper, we adopt Denoising Diffusion Probabilistic Models (DDPM) [5] as the sampling method, where $p\left(\boldsymbol{\tau}^i|\boldsymbol{\tau}^0\right) = \mathcal{N}\left(\sqrt{\bar{\alpha}_i}\boldsymbol{\tau}^0, (1-\bar{\alpha}_i)\,\boldsymbol{I}\right)$, $\bar{\alpha}_i := \prod_{j=1}^{i} \alpha_i$, $\alpha_i := 1 - \beta_i$ and $\{\beta_i\}$ is a sequence of positive noise scales $0 < \beta_1, \beta_2, \ldots, \beta_N < 1$.

B. Linear Temporal Logic (LTL)

Given a finite set of propositional symbols $\mathcal{P}$, the formula set Ψ of LTL_f contains formulas recursively defined in Backus-Naur form as follows [11], [12]:

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi \,\mathsf{U}\, \psi,$$

where $p \in \mathcal{P}$ and $\varphi, \psi \in \Psi$. Intuitively, the formula $\bigcirc\varphi$ (*next* φ) is satisfied if φ is satisfied at the next time step. $\varphi \,\mathsf{U}\, \psi$ (φ *until* ψ) is satisfied if φ is satisfied until ψ is satisfied, and ψ is satisfied by the end of the sequence. From these, other commonly used logical connectives and temporal operators can be defined according to the following equivalences: $\varphi \vee \psi = \neg\left(\neg\varphi \wedge \neg\psi\right)$, $\Diamond\varphi = \mathsf{true}\,\mathsf{U}\,\varphi$ (*eventually* φ) and $\Box\varphi = \neg\Diamond\left(\neg\varphi\right)$ (*always* φ). The symbols true and false can also be in the formula set defined by $\mathsf{true} = \varphi \vee \neg\varphi$ and $\mathsf{false} = \neg\mathsf{true}$.

In contrast to propositional logic, these formulas are evaluated over finite sequences of observations $\boldsymbol{\sigma} = \langle\sigma_0, \sigma_1, \sigma_2, \ldots \sigma_T\rangle$ (*i.e.*, *truth assignments* to the propositional symbols in $\mathcal{P}$), where $\sigma_t \in \{0,1\}^{|\mathcal{P}|}$ and $\sigma_{t,p} = 1$ iff proposition $p \in \mathcal{P}$ is satisfied at time step t. true (false) is always satisfied (not satisfied) by any assignment. Formally, $\boldsymbol{\sigma}$ *satisfies* φ at time $t \geq 0$, denoted by $\langle\boldsymbol{\sigma}, t\rangle \models \varphi$, as follows:

- $\langle\boldsymbol{\sigma}, t\rangle \models p$ iff $\sigma_{t,p} = 1$, where $p \in \mathcal{P}$
- $\langle\boldsymbol{\sigma}, t\rangle \models \neg\varphi$ iff $\langle\boldsymbol{\sigma}, t\rangle \not\models \varphi$
- $\langle\boldsymbol{\sigma}, t\rangle \models (\varphi \wedge \psi)$ iff $\langle\boldsymbol{\sigma}, t\rangle \models \varphi$ and $\langle\boldsymbol{\sigma}, t\rangle \models \psi$
- $\langle\boldsymbol{\sigma}, t\rangle \models \bigcirc\varphi$ iff $\langle\boldsymbol{\sigma}, t+1\rangle \models \varphi$
- $\langle\boldsymbol{\sigma}, t\rangle \models \varphi \,\mathsf{U}\, \psi$ iff $\exists t_2 \in [t, T]$ s.t. $\langle\boldsymbol{\sigma}, t_2\rangle \models \psi$ and $\forall t_1 \in [t, t_2)$, $\langle\boldsymbol{\sigma}, t_1\rangle \models \varphi$

A sequence $\boldsymbol{\sigma}$ is then said to *satisfy* φ, *i.e.*, $\boldsymbol{\sigma} \models \varphi$, iff $\langle\boldsymbol{\sigma}, 0\rangle \models \varphi$.

III. METHOD

In this section, we describe our primary contribution, LTLDOG, a diffusion-based framework for generating trajectories that satisfy LTL$_f$ formulae. We first discuss how to conditionally sample using diffusion models, followed by how LTL$_f$ formulae can be used to guide the diffusion process.

A. Conditional Sampling in Diffusion Models

Given a trained score function from (1) such that $s_\theta \approx \nabla_{\boldsymbol{\tau}^i} \log p_i\left(\boldsymbol{\tau}^i\right)$, a diffusion model denoises samples according to the distribution $p_\theta\left(\boldsymbol{\tau}^{i-1}|\boldsymbol{\tau}^i\right) = \mathcal{N}\left(\frac{1}{\sqrt{\alpha_i}}\left(\boldsymbol{\tau}^i + (1-\alpha_i)\, s_\theta\left(\boldsymbol{\tau}^i, i\right)\right), (1-\alpha_i)\, \boldsymbol{I}\right)$ starting from a Gaussian prior $\boldsymbol{\tau}^N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. For example, DIFFUSER [1] samples a trajectory $\boldsymbol{\tau}^0$ from a diffusion model, which an agent then executes. However, this original sampling process is unable to control detailed properties of generated context.

Here, we are interested in sampling trajectories that satisfy both the final goal and the specified instructions encoded as an LTL$_f$ formula φ that is provided during deployment. In other words, we aim to sample trajectories under an objective function $\mathcal{J}_\varphi\left(\boldsymbol{\tau}^0|g\right)$. For example, $\mathcal{J}$ can have a high value if the events induced by $\boldsymbol{\tau}^0$ satisfy φ and have a low value otherwise.

Formally, given a set of atomic propositions $\mathcal{P}$, the assignments for $\boldsymbol{\tau}$ are given by a labeling function $L : \mathcal{S}\times\mathcal{A} \to 2^{|\mathcal{P}|}$, where each timestep of $\boldsymbol{\tau}$ induces an assignment to the propositional symbols in $\mathcal{P}$. For example, in navigation task, $\mathcal{P}$ can represent multiple regions to avoid and L is a function indicating whether $(\mathbf{s}_t, \mathbf{a}_t)$ in $\boldsymbol{\tau}$ are inside these regions or not. We will slightly abuse notation and write $\boldsymbol{\tau} \models \varphi$ to indicate that $\boldsymbol{\tau}$'s assignments $\boldsymbol{\sigma}$ satisfy a LTL$_f$ formula.

We aim to sample from the posterior $p_0\left(\boldsymbol{\tau}^0|\boldsymbol{\tau}^0 \models \varphi, g\right)$ with the diffusion model as the prior. In this work, the constraint of goal state conditioning g can be either achieved by inpainting (similarly on the start state $\mathbf{s}_0$) or implicitly encoded in the dataset, which does not require separate modeling in conditional sampling. Therefore, we mainly target the posterior $p_0\left(\boldsymbol{\tau}^0|\boldsymbol{\tau}^0 \models \varphi\right)$. Let y_0 denote a binary random variable indicating the likelihood of $\boldsymbol{\tau}^0$ satisfying φ. Hence, the denoising process requires a score function conditioned on $y_0 = 1$ and by Bayes' rule: $\nabla_{\boldsymbol{\tau}^i} \log p_i\left(\boldsymbol{\tau}^i|y_0 = 1\right) = \nabla_{\boldsymbol{\tau}^i} \log p_i\left(\boldsymbol{\tau}^i\right) + \nabla_{\boldsymbol{\tau}^i} \log p_i\left(y_0 = 1|\boldsymbol{\tau}^i\right)$. The first term on the right-hand side has been learned by a neural network in diffusion. However, the latter term requires an integration over all possible values of $\boldsymbol{\tau}^0$: $p_i\left(y_0|\boldsymbol{\tau}^i\right) = \int p\left(y_0|\boldsymbol{\tau}^0\right) p_i\left(\boldsymbol{\tau}^0|\boldsymbol{\tau}^i\right) \mathrm{d}\boldsymbol{\tau}^0$. We consider the plug-and-play conditional generation setting and approximating this integration with sample estimation [13], *e.g.*, point estimation with $p_i\left(y_0|\boldsymbol{\tau}^i\right) \approx p\left(y_0|\hat{\boldsymbol{\tau}}^0\right)$ where the noiseless trajectory $\hat{\boldsymbol{\tau}}^0$ is estimated via Tweedie's formula [14] $\hat{\boldsymbol{\tau}}^0 = \frac{1}{\sqrt{\bar{\alpha}_i}}\left(\boldsymbol{\tau}^i + (1-\bar{\alpha}_i)\, \nabla_{\boldsymbol{\tau}^i} \log p_i\left(\boldsymbol{\tau}^i\right)\right)$. We model the likelihood term as $p\left(y|\boldsymbol{\tau}\right) = Ze^{\mathbb{1}[y=\mathbb{1}[\boldsymbol{\tau}\models\varphi]]}$, where Z is a normalizing constant and $\mathbb{1}\left[\cdot\right]$ is the indicator function. Putting the above elements together, the conditional score function can be computed as follows,

$$\begin{aligned}\nabla_{\boldsymbol{\tau}^i} \log p_i\left(y_0|\boldsymbol{\tau}^i\right) &\approx \nabla_{\boldsymbol{\tau}^i} \log p\left(y_0|\hat{\boldsymbol{\tau}}^0\right) \\ &= \nabla_{\boldsymbol{\tau}^i} \log\left(Ze^{\mathbb{1}[y=\mathbb{1}[\hat{\boldsymbol{\tau}}^0\models\varphi]]}\right) \\ &= \nabla_{\boldsymbol{\tau}^i} \mathbb{1}\left[y = \mathbb{1}\left[\hat{\boldsymbol{\tau}}^0 \models \varphi\right]\right].\end{aligned} \tag{2}$$

Unfortunately, both the indicator function and the satisfaction evaluation (performed by techniques like model checking with finite automata) are non-differentiable — this prohibits application in the gradient-based sampling process of diffusion models. We address this problem in the next subsection.

B. Differentiable Evaluation of LTL$_f$

Our key approach is to "soften" the satisfaction evaluation. To make the evaluation differentiable, we modify our formula evaluator to output positive real values if a trajectory satisfies the LTL$_f$ formula, and negative real values otherwise. Instead of using a binary labeling functions, we assume real valued assignments of atomic propositions can be obtained through computation on the generated trajectories. As a specific example, consider a navigation task where a robot has to avoid obstacles; one can determine the assignment for proposition p at timestep t using the Euclidean distance between $\mathbf{s}_t$ and the centers of the region $\mathbf{c}_p$ (assuming a circular shape). A positive value of $\ell(p,t) = r_p - \|\mathbf{s}_t - \mathbf{c}_p\|_2$, where r_p is the radius of the circle, indicates a true assignment. Consequently, the labeling function for the entire trajectory $L : (\mathcal{S}\times\mathcal{A})^{T+1} \to \mathbb{R}^{|\mathcal{P}|\times(T+1)}$ is differentiable and the assignments are $\boldsymbol{\sigma} = L(\boldsymbol{\tau})$.

A binary version of $\boldsymbol{\sigma}$ through a sign function $\mathrm{sgn}(\boldsymbol{\sigma})$ satisfies the definition of *satisfy* defined in Section II-B, but the sign operation breaks differentiability. As such, we employ a formula evaluator $f_t\left(\varphi, \boldsymbol{\sigma}_{t:T}\right) : \Psi \times \mathbb{R}^{|\mathcal{P}|\times(T-t+1)} \to \mathbb{R}$ to check satisfaction [15], with positive values implying $\langle\boldsymbol{\sigma}, t\rangle \models \varphi$, similar to signal temporal logic [16]. As such, f is differentiable with the evaluation process defined as follows:

- $f_t(\mathsf{true}, \boldsymbol{\sigma}) = +\infty$
- $f_t(\mathsf{false}, \boldsymbol{\sigma}) = -\infty$
- $f_t(p, \boldsymbol{\sigma}) = \boldsymbol{\sigma}_{t,p}$
- $f_t(\neg\varphi, \boldsymbol{\sigma}) = -f_t(\varphi, \boldsymbol{\sigma})$
- $f_t(\varphi \wedge \psi, \boldsymbol{\sigma}) = \min^\gamma\left\{f_t(\varphi, \boldsymbol{\sigma}), f_t(\psi, \boldsymbol{\sigma})\right\}$
- $f_t(\varphi \vee \psi, \boldsymbol{\sigma}) = \max^\gamma\{f_t(\varphi, \boldsymbol{\sigma}), f_t(\psi, \boldsymbol{\sigma})\}$
- $f_t(\bigcirc\varphi, \boldsymbol{\sigma}) = f_{t+1}(\varphi, \boldsymbol{\sigma})$
- $f_t(\Box\varphi, \boldsymbol{\sigma}) = \min^\gamma\{f_{t:T}(\varphi, \boldsymbol{\sigma})\}$
- $f_t(\Diamond\varphi, \boldsymbol{\sigma}) = \max^\gamma\{f_{t:T}(\varphi, \boldsymbol{\sigma})\}$
- $f_t(\varphi \,\mathsf{U}\, \psi, \boldsymbol{\sigma}) = \min^\gamma\{f_{t:k}(\varphi, \boldsymbol{\sigma}), f_t\left(\Diamond\psi, \boldsymbol{\sigma}\right)\}$, where $k \geq t$ is the smallest integer s.t. $f_k(\psi, \boldsymbol{\sigma}) > 0$

Note that the $\min$ and $\max$ functions are likewise "soft" to maintain differentiability. To reduce clutter, we have omitted the subscript t when $t = 0$. With these operations, f maintains *quantitative semantics*, which preserves the relative values between different $\boldsymbol{\sigma}$ such that trajectories with larger margin to a satisfying assignment have larger values.

Given L and f, we can obtain a differentiable score function by replacing the likelihood term in (2) with $p\left(y|\boldsymbol{\tau}\right) = Ze^{(2y-1)f(\varphi,L(\boldsymbol{\tau}))}$. The conditional score is then

$$\begin{aligned}\nabla_{\boldsymbol{\tau}^i}\log p_i\left(y_0=1|\boldsymbol{\tau}^i\right) &\approx \nabla_{\boldsymbol{\tau}^i}\log\left(Ze^{f(\varphi,L(\hat{\boldsymbol{\tau}}^0))}\right)\\ &= \nabla_{\boldsymbol{\tau}^i} f\left(\varphi, L\left(\hat{\boldsymbol{\tau}}^0\right)\right).\end{aligned} \tag{3}$$

This approximation of the conditional score function can be directly used with a pre-trained diffusion model to sample trajectories conditioned on an LTL$_f$ formula. We call this method LTLDoG-S, since it performs the above posterior sampling in the reverse process (Algorithm 1). The gradient ascent step is controlled by a stepsize $\{\zeta_i\}_{i=1}^N$; in practice, the stepsize for each denoise step can be adaptive such that $\boldsymbol{\tau}^{i-1}$ remains valid according to the formula evaluator $f\left(\varphi, L\left(\hat{\boldsymbol{\tau}}^0\right)\right)$.

Algorithm 1 LTL$_f$ Planning with Posterior Sampling

Require: φ, N, s_θ, $\{\zeta_i\}_{i=1}^N$
$\boldsymbol{\tau}^N \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$
for $i = N-1$ **to** 0 **do**
  $\hat{s} \leftarrow s_\theta\left(\boldsymbol{\tau}^i, i\right)$
  $\hat{\boldsymbol{\tau}}^0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}\left(\boldsymbol{\tau}^i + (1-\bar{\alpha}_i)\,\hat{s}\right)$
  $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$
  $\boldsymbol{\tau}^{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{\tau}^i + \frac{\sqrt{\bar{\alpha}_{i-1}}(1-\alpha_i)}{1-\bar{\alpha}_i}\hat{\boldsymbol{\tau}}^0 + \sqrt{1-\alpha_i}\boldsymbol{\epsilon}$
  $\boldsymbol{\tau}^{i-1} \leftarrow \boldsymbol{\tau}^{i-1} + \zeta_i \nabla_{\boldsymbol{\tau}^i} f\left(\varphi, L\left(\hat{\boldsymbol{\tau}}^0\right)\right)$
end for
return $\hat{\boldsymbol{\tau}}^0$

C. Classifier Guidance over LTL$_f$

One limitation of LTLDoG-S is that it necessitates a known (and differentiable) expression for the formula evaluator $f\left(\varphi, L\left(\hat{\boldsymbol{\tau}}^0\right)\right)$. This requirement can be challenging to meet in scenarios where the truth assignments of propositions are uncertain, for example, when the ground-truth physical dynamics are unknown. Here, we circumvent this problem by employing classifier guidance using a trained formula evaluator.

We propose a variant of LTLDoG with *regressor guidance*, which we abbreviate as LTLDoG-R. Using an ℓ_2 loss, we train a neural network to predict the satisfaction values from noisy trajectories conditioned on LTL$_f$ instructions. In other words, our neural network approximates the conditional score function $\nabla_{\boldsymbol{\tau}^i}\log p_i\left(y_0=1|\boldsymbol{\tau}^i\right)$. Note that in contrast to learning a binary classifier for trajectory satisfaction, we apply the labeling function in Section III-B on the dataset and associate with each trajectory its objective value $\mathcal{J}_\varphi\left(\boldsymbol{\tau}^0\right)$. We conduct ablation study in Section V-D to show that using real values performs better than using binary labels.

To generalize over different LTL$_f$, the neural network takes both noisy trajectory and LTL$_f$ formula embedding as input. Multiple methods exist for embedding LTL$_f$ formulae. For instance, one can use Graph Neural Networks (GNNs) [17], [18] to embed the tree representation of an LTL$_f$ formula directly. Alternatively, the deterministic finite-state automaton (DFA) [19], [20] associated with the formulae can be embedded [21]. In this work we employ LTL$_f$ tasks from [22] and embed the directed graph of an LTL$_f$ formula using the Relational Graph Convolutional Network (R-GCN) [23], which can generalize to LTL$_f$ formulae with same template structure. The model that approximates the score function $s_\phi(\phi, \boldsymbol{\tau}^i, i) \approx \nabla_{\boldsymbol{\tau}^i}\log p_i\left(y_0|\boldsymbol{\tau}^i\right)$ after training can be plugged into the conditional reverse process using regressor guidance as summarized in Algorithm 2.

Algorithm 2 LTL$_f$ Planning with Regressor Guidance

Require: φ, N, s_θ, s_ϕ, $\{\zeta_i\}_{i=1}^N$
$\boldsymbol{\tau}^N \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$
for $i = N-1$ **to** 0 **do**
  $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$
  $\boldsymbol{\tau}^{i-1} \leftarrow \frac{1}{\sqrt{\alpha_i}}\left(\boldsymbol{\tau}^i + (1-\alpha_i)s_\theta\left(\boldsymbol{\tau}^i, i\right)\right) + \sqrt{1-\alpha_i}\boldsymbol{\epsilon}$
  $\boldsymbol{\tau}^{i-1} \leftarrow \boldsymbol{\tau}^{i-1} + \zeta_i s_\phi(\phi, \boldsymbol{\tau}^i, i)$
end for
return $\hat{\boldsymbol{\tau}}^0$

IV. Related Work

LTLDoG builds upon prior work in diffusion-based planning and symbolic reasoning using LTL$_f$ for robotics. In the following, we give a brief overview of related work.

Learning and Planning under LTL$_f$. As an expressive language for specifying high-level planning requirements [24]–[26], LTL$_f$ has been extensively used in various robotic tasks to express temporally extended goals [27], [28]. These methods usually require the information about the environment's dynamics, *e.g.*, a model or an abstraction, to effectively plan under a given formula. Reinforcement learning agent learn in a model-free way under LTL$_f$ objectives or constraints [29]–[31] with the ability to generalize over different formulae [22]. However, these methods operate agent in an online manner via trial and error, which can lead to expensive or even unsafe interactions.

Planning and Policy Learning with Diffusion. Recent diffusion-based planning methods are flexible that only rely on offline datasets without access or interaction to environments. They have been successfully applied to long-horizon planning problems by generating states or actions for control [1]–[3], but not tasks with test-time temporal requirements. Recent work has looked into safety critical tasks *e.g.*, the aforementioned CBF-based methods [7], [8]. As discussed above, these methods were designed for static safety criteria, and the lack the ability to satisfy temporally extended behaviors. Our work inherits the advantages of diffusion based methods and can fulfill LTL$_f$ requirements.

Inverse Problems in Diffusion. Our proposed method formulates conditional measurement under LTL$_f$ with differentiable loss function using the unnormalized likelihood [13], [32] for posterior sampling and can guide the sampling process in a plug-and-play fashion. The most popular methods to guide diffusion models during inference is classifier guidance [33] and classifier-free guidance [34]. However, these methods cannot be applied in a plug-and-play fashion for new conditioning factors. Our work is also related to inverse task that

(a) Studio-like Lab. (b) Office room. (c) Unitree Go2.

Fig. 2. Real world environments for quadruped robot navigation.

TABLE I
PERFORMANCE ON AVOIDANCE TASKS IN MAZE2D

Method\Perf.	Satisfaction rate[1](%) ↑		Reward (UnCon)[2]↑
	Planning	Rollout	
DIFFUSER	9.5±3.1	11.0±1.7	142.2±5.1
SAFEDIFFUSER	99.4±0.9	12.3±3.5	135.8±5.2
LTLDOG-S	99.0±0.8	73.0±3.0	97.3±2.8
LTLDOG-R	98.8±0.8	**92.0±1.4**	127.1±5.1

[1] Mean and standard deviation calculated from 10 groups of tests. Each test contains 100 trials, where a trajectory is labeled as either satisfied or not satisfied in each trial. Best result during rollout is highlighted. Same for other tables of Maze2d.
[2] Unconstrained rewards do *not* take unsafe penalties into account.

TABLE II
RESULTS OF AVOIDANCE TASK IN PUSHT

Method\Perf.	Satisfaction rate(%) ↑	Overlap Score[1]↑
DIFFUSION POLICY	34.8±18.0	0.941±0.0584
LTLDOG-S	**85.6±13.1**	0.890±0.0647
LTLDOG-R	**85.6±12.5**	0.842±0.0985

[1] measures the final overlap of the T block and target area (min: 0.0, max: 1.0).

infers a posterior distribution over data given a measurement. Inverse methods [35]–[38] do not require training a conditional model in diffusion and can directly leverage pre-trained neural networks.

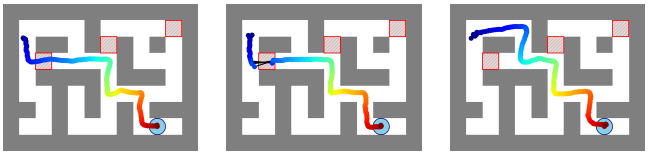
V. EXPERIMENTS

Our experiments focus on testing LTLDOG's ability to handle static and temporal safety constraints. We first briefly describe the simulated and real environments, and baseline data-driven methods. Then we report empirical results on benchmark environments and demonstrate LTLDOG's applicability in real world tasks through a case study on a quadruped robot (Fig. 2). We conclude with a brief ablation study and analysis. Due to space restrictions, we focus on conveying our main results. More details of the environments, implementations and analysis can be found in the online supplementary material[1].

A. Experimental Setup

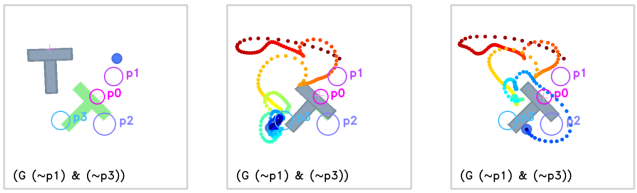
Environments. We evaluate methods in two simulation benchmark environments (Maze2d [1] and PushT [3]) and demonstrate in two real indoor rooms. Maze2d (Fig. 3 and 5) presents challenging long-horizon navigation tasks, where state-of-the-art offline algorithms fail to learn a reliable goal-reaching policy. The atomic propositions are determined by

[1]https://github.com/clear-nus/ltldog



(a) DIFFUSER. (b) SAFEDIFFUSER. (c) LTLDOG-S.

Fig. 3. Examples of safe planning in Maze2d-Large. There are three unsafe blocks (red squares, labeled p_L, p_M, p_R from left to right) that need to be avoided during navigation to the goal (shaded circle). The LTL$_f$ constraint for this task is $\varphi = \Box\neg(p_L \wedge p_M \wedge p_R)$. (a) Trajectories from DIFFUSER ignore safety and can violate the specified constraints. (b) SAFEDIFFUSER produces discontinuous trajectories. (c) Our LTLDOG is able to plan trajectories that detours around the obstacles to successfully arrive at the goal.



(a) PushT environment. (b) DIFFUSION POLICY. (c) LTLDOG-S.

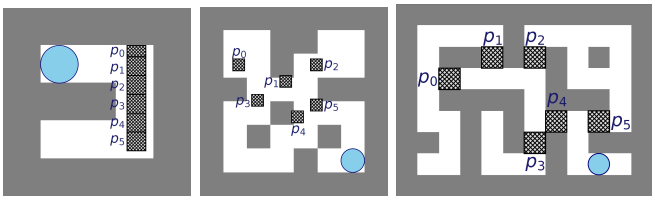
Fig. 4. Results of safe control in PushT. (a) A robot arm's end effector (circles filled in blue) should manipulate the T block (gray) to a goal pose (green), and avoid entering unsafe regions (hollow circles marked with pX), specified by an LTL$_f$ formula (text in black). In this example, the LTL$_f$ specifies the end effector should never enter regions $p1$ (purple) and $p3$ (cyan). (b) The actions generated and executed by DIFFUSION POLICY do not satisfy the LTL$_f$ formula. (c) In contrast, LTLDOG-S guides the diffusion to avoid $p1$ and $p3$, yet still completes the manipulation task.

the occurrence of events when the agent is inside key regions in the maze (for avoidance and visitation). The PushT task (Fig. 4) requires manipulation of a T block through interaction with a controllable mover. In our experiments, the mover is constrained to visit specific regions and avoid others. Our real-world experiments involve two indoor environments: a lab designed to mimic a studio apartment, and an office environment (Fig. 2).

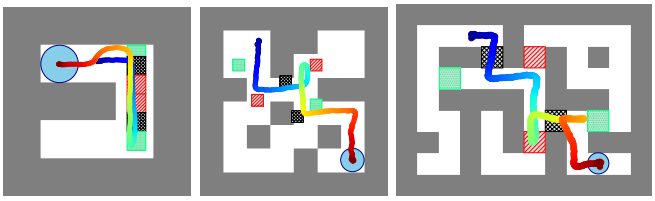
Compared methods. Our work involves trajectory generation by learning from an offline dataset and as such, we compare against data-driven planning methods. DIFFUSER and DIFFUSION POLICY are state-of-the-art methods for sampling viable plans but without any guarantees over external constraints. To evaluate how well LTLDOG enforces safety specifications, we compare with SAFEDIFFUSER, a safe planning diffusion model using CBFs. However, note that SAFEDIFFUSER cannot handle temporal constraints or instructions; to our knowledge, our work is the first data-driven method to handle both static and temporal constraints. As such, there is no direct comparison baseline. We analyze the differences between the two variants of our method, LTLDOG-S and LTLDOG-R.

B. Comparative Analysis of Methods

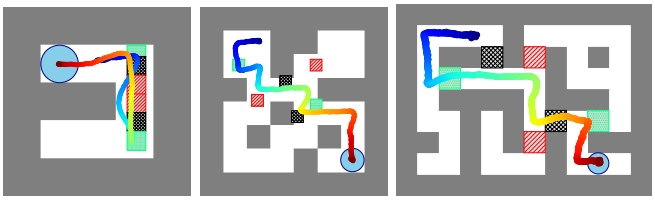
Can LTLDOG achieve safe planning for static constraints? Our results indicate that yes, LTLDOG is better able to generate trajectories that satisfy given region-avoidance constraints relative to existing methods. In both the Maze2D and PushT benchmarks, LTLDOG achieves high success rates (in both



(a) Regions for evaluation of atomic propositions in $\mathcal{P}$.



(b) DIFFUSER.



(c) Ours (LTLDOG-S).

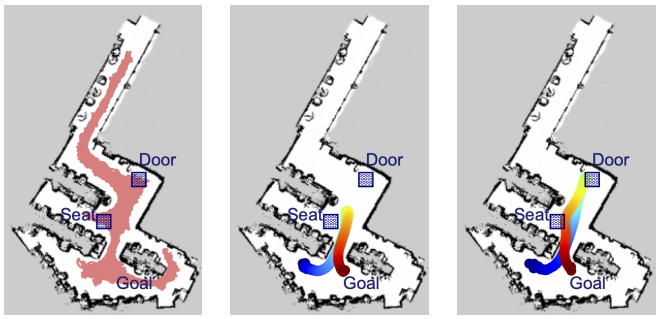
Fig. 5. Temporal Constraints in Maze2D. (a) Each maze has 6 non-overlapping regions. Agents are requested to visit some of these blocks under different temporally-extended orders. (b) and (c) show generated trajectories under $\varphi = \neg p_3 \,\mathsf{U}\, (p_5 \wedge (\neg p_2 \,\mathsf{U}\, p_0))$. Our method can satisfy $\neg$ propositions (red zones) before reaching the green regions.

planning and rollout), without severely compromising reward accumulation (Tables I and II). Lower total rewards are expected since safe paths are typically longer and rollout scores are generally lower as the low-level controller may not exactly follow the diffused trajectory. Qualitatively, Fig. 3 shows that LTLDOG is able to generate safe trajectories in Maze2D, whilst SAFEDIFFUSER is limited to "local" deviations and fails to find paths that detour around unsafe regions. Results in PushT are consistent with Maze2D; Fig. 4 shows that LTLDOG performs the orientation task without entering unsafe regions.

Can LTLDOG satisfy static and temporal constraints, and generalize to novel LTL_f formulae? Tables III and IV show the performance of the compared methods on both training and test LTLs in Maze2d and PushT environments. To elaborate, we follow the LTL_f specifications in [22] and adopt the `Until` sampler to generate random LTL_fs (200 for Maze2d and 36 for PushT) that contain different visitation sequences and unsafe regions. The training set has 80% of all LTL_fs and the rests are used as test set. Atomic propositions consist of 6 regions in Maze2d (Fig. 5(a), *i.e.*, $p_0, p_1, \ldots, p_5$) and 4 regions in PushT (Fig. 4(a)).

Results reveal that both LTLDOG-S and LTLDOG-R achieve significantly higher success rates than DIFFUSER/DIFFUSION POLICY. The baselines have a non-zero performance as some generated LTL_f formulae are trivial to satisfy at some start locations. Also note that some specifications may be impossible to satisfy given the physical locations of the agent, walls, and propositional regions in the maze. In these cases, planned trajectories will violate constraints during rollout, causing a drop in the satisfaction rate. Fig. 5 shows examples of trajectories generated in Maze2d.

(a) Lab map and dataset coverage. (b) DIFFUSER. (c) Ours.



(d) Office map and dataset coverage. (e) DIFFUSER. (f) Ours.

Fig. 6. Results in real world rooms. The instructed LTL_f is $\varphi = \neg\text{Passageway}\,\mathsf{U}\,\text{Kitchen}$ for lab (first row) and $\varphi = \Diamond(\text{Door} \wedge \Diamond\text{Seat})$ for office (second row). In the lab task the robot has to first unload in the kitchen area after entering the door before being allowed to go to the goal. When loaded, the robot is prohibited from going through the narrow passageway where people sit on the side. In the office task, the robot has to first visit the seat, followed by the door, before the goal.

C. Real Robot Case Study.

We show that LTLDOG can plan for a robot dog (Unitree Go2 Edu) given LTL_f instructions in two real-world navigation environments — a lab that mimics a studio apartment and an office room. Training of diffusion models was performed in simulation using Gazebo and ROS1 using LIDAR scanned maps (Fig. 6(a) and 6(d)). Note that the training trajectories do not require running an oracle policy to satisfy many different LTL_f formulae; we simply made the robot navigate to randomly sampled goals using the global planner and TEB Local Planner [39] from the ROS1 Navigation stack (Fig. 6(a) and 6(d)).

To test on potential constraints, we queried GPT-4 for LTL_f formulae representing meaningful robotic tasks including obstacle avoidance and sequential navigation. We used 4 kinds of LTL_fs from the generated results, *i.e.*, $\Box\neg p_0$, $\Diamond p_0 \wedge \Diamond p_1$, $\Diamond(p_0 \wedge \Diamond p_1)$ and $\neg p_1 \,\mathsf{U}\, p_0$. The first LTL_f corresponds to an obstacle avoidance task where the robot should never visit a specific region. For example, $\Box\neg$Seat means the robot should never enter the Seat region (Fig 6(d)). The remaining three LTL_f formulae represent: 1) visiting all regions at least once; 2) visiting regions in a specific sequence; 3) avoiding a specific region until another has been visited. Some example regions designed in our real environments and start/goal locations are shown in Fig 6.

In total, 96 trials were executed on the real robot, involving 12 trajectories (6 for baseline DIFFUSER and 6 for our method LTLDOG-S) with varying regions and different LTL_f

TABLE III
PERFORMANCE ON DIFFERENT LTL$_f$S IN MAZE2D.

Environment	Method\Performance	Training LTL$_f$s			Testing LTL$_f$s		
		Satisfaction rate (%) ↑		Reward (UnCon) ↑	Satisfaction rate (%) ↑		Reward (UnCon) ↑
		Planning	Rollout		Planning	Rollout	
U-Maze (Horizon 256)	DIFFUSER	31.1±0.5	31.0±0.5	33.5±2.7	33.9±0.5	34.1±0.6	35.6±0.3
	LTLDOG-S	83.8±0.2	**57.6±1.3**	31.3±1.2	82.7±0.3	**56.6±0.9**	32.8±0.5
	LTLDOG-R	56.3±0.4	51.3±0.9	31.5±0.2	57.7±0.4	52.1±0.3	32.3±0.7
Medium (Horizon 384)	DIFFUSER	15.0±0.7	13.4±0.6	84.8±0.3	11.6±1.4	10.1±1.2	84.8±0.5
	LTLDOG-S	77.9±5.7	31.8±2.6	53.1±5.2	68.4±6.7	28.7±3.5	50.5±4.7
	LTLDOG-R	51.8±1.8	**39.5±1.6**	57.3±0.2	43.3±4.4	**30.6±1.9**	57.7±0.1
Large (Horizon 512)	DIFFUSER	13.5±0.4	12.8±0.1	76.3±0.1	11.6±2.3	11.5±1.7	77.8±3.9
	LTLDOG-S	73.8±2.4	32.6±1.4	42.3±5.0	66.6±2.7	24.9±1.7	40.9±4.4
	LTLDOG-R	66.9±0.6	**47.4±0.8**	54.6±1.3	57.5±2.3	**39.0±2.9**	54.5±3.9

TABLE IV
GENERALIZATION TO DIFFERENT LTL$_f$ FORMULAE IN PUSHT

Method\Performance	LTL$_f$ Set	Satisf. rate (%) ↑	Score ↑
DIFFUSION POLICY	Training	22.9±8.0	0.354±0.153
	Test	30.7±13.9	0.371±0.177
LTLDOG-S	Training	28.2±8.33	0.290±0.115
	Test	43.0±17.0	0.299±0.145
LTLDOG-R	Training	**69.3±9.90**	0.292±0.121
	Test	**66.0±20.8**	0.340±0.168

TABLE V
RESULTS OF ACHIEVING GOALS AND LTL$_f$ IN REAL-WORLD TASKS.

Environment	Method\Performance	Satisfaction rate (%) ↑	
		Goal	LTL$_f$
Lab	DIFFUSER	100.00	0.00
	LTLDOG-S	91.67	**91.67**
Office	DIFFUSER	100.00	0.00
	LTLDOG-S	95.83	**95.83**

formulae in each room. The overall satisfaction rate of all raw generated trajectories in *simulation* is $85.8 \pm 14.0\%$ (*c.f.* baseline $2.9 \pm 5.4\%$). For each specific formula, a sample was selected for real-world execution based on their feasibility. The results in TABLE V show that LTLDOG has a high satisfaction rate compared to DIFFUSER; Fig. 6(c) and 6(f) illustrate trajectories for qualitative comparison.

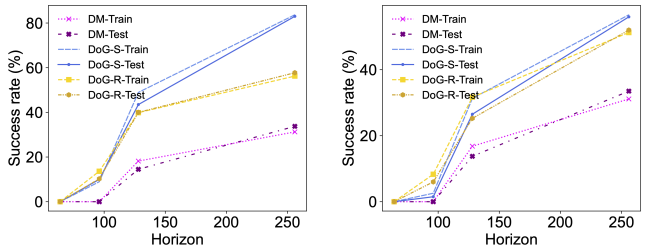
D. Ablation Study and Analysis

Ablation study. Unlike classifier guidance, where each trajectory is labelled as *satisfy* or *not satisfy*, we leverage the continuous values from our formula evaluator (as described in Section III-C) and train a regressor guidance network. Although classifier guidance improves over the vanilla DIFFUSER, it achieves lower performance than LTLDOG-R (Table VI). This comparison supports the notion that soft labels improves the guidance neural network; we posit using real values provides richer information in terms of how *well* the trajectory satisfies a given LTL$_f$ formula.

TABLE VI
ABLATION STUDY – BINARY CLASSIFIER GUIDANCE

Method\Performance	LTL Set	Satisfaction rate (%) ↑		Reward (UnCon) ↑
		Planning	Rollout	
DIFFUSER	Training	31.1±0.5	31.0±0.5	32.5±2.7
	Test	33.9±0.5	34.1±0.6	35.6±0.3
Classifier guidance[1]	Training	41.1±0.6	40.6±0.7	33.9±0.9
	Test	40.0±0.8	41.8±1.1	35.3±0.3
LTLDOG-R	Training	56.3±0.4	**51.3±0.9**	31.5±0.2
	Test	57.7±0.4	**52.1±0.3**	32.3±0.7

[1] The classifier guidance method only leverages *binary labels* for LTL satisfaction checking in Maze2d U-Maze.



(a) Planning. (b) Rollout.

Fig. 7. Performance in Maze2d U-Maze with different lengths of trajectory. DM stands for DIFFUSER and DoG-S/DoG-R are our methods.

Analysis on horizon. Fig. 7 demonstrates that planning with a longer horizon leads to improved performance in terms of LTL$_f$ satisfaction. This improvement is attributed to the fact that LTL$_f$ instructions often require a longer sequence of steps in a path compared to mere goal navigation, *e.g.*, visiting a specific region before reaching the goal.

VI. CONCLUSION, DISCUSSION AND FUTURE WORK

In this work, we presented LTLDOG, an approach towards generating safe trajectories that comply with LTL$_f$ specifications at test time. Within our overall scheme, we presented two methods: LTLDOG-S guides the sampling process under any LTL$_f$ formula while LTLDOG-R uses a trained model that we show generalizes to new formulae with similar structure. To our knowledge, this work is the first that successfully incorporates model checking using a formal language with diffusion models for safe planning. Notably, LTLDOG does not require data collection for each potential LTL$_f$ instruction; rather, we control the sampling process during diffusion using "soft" model checking to generate new trajectories using existing information provided by the training dataset.

Limitations and Future Work. LTLDOG is a step towards trustworthy trajectory planning using generative models. There are several areas where LTLDOG can be improved. Similar to

other diffusion models, LTLDoG is generally unable to generate realistic trajectories when the context (environment/goal) is far from the training distribution. As such, the dataset should preferably contain trajectories with a variety of complex behaviors that can potentially satisfy different test-time LTL_f formulae. It would be interesting to develop methods to ensure the sampling process of LTLDoG-S adheres to the data manifold. Additionally, LTLDoG plans in an open-loop fashion, where the entire trajectory is generated conditioned on the constraints. This is mainly because the evaluation of an LTL_f formula depends on the entire trajectory. We plan to further explore planning using receding horizon control using partial evaluations on an incomplete trajectory. Finally, diffusion models usually require large amounts of training data and many diffusion steps during inference. Recent work on interpolant diffusion methods [40] leverages source policies to reduce data and computation costs. We aim to explore how integrating this approach with conditional sampling using LTL_f can enhance the generation of safe trajectories.

Acknowledgements

This research is supported by A*STAR under its National Robotics Programme (NRP) (Award M23NBK0053). The authors would also like to acknowledge partial support from a Google South Asia & Southeast Asia Award and from the National Research Foundation, Singapore under its Medium Sized Center for Advanced Robotics Technology Innovation.

References

[1] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Int. Conf. Mach. Learn.*, vol. 162, 2022, pp. 9902–9915.

[2] A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. S. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision making?" in *Int. Conf. Learn. Representations*, 2023.

[3] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proc. Robot.: Sci. and Syst. (RSS)*, 2023.

[4] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[6] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Int. Conf. Learn. Representations*, 2021.

[7] W. Xiao, T.-H. Wang, C. Gan, and D. Rus, "SafeDiffuser: Safe planning with diffusion probabilistic models," *arXiv preprint arXiv:2306.00148*, 2023.

[8] N. Botteghi, F. Califano, M. Poel, and C. Brune, "Trajectory generation, control, and safety with denoising diffusion probabilistic models," *arXiv preprint arXiv:2306.15512*, 2023.

[9] A. Pnueli, "The temporal logic of programs," in *18th Annu. Symp. Found. Comput. Sci.*, 1977, pp. 46–57.

[10] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, 2011.

[11] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[12] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017, vol. 89.

[13] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," in *Int. Conf. Learn. Representations*, 2023.

[14] B. Efron, "Tweedie's formula and selection bias," *J. Amer. Statistical Assoc.*, vol. 106, no. 496, pp. 1602–1614, 2011.

[15] Z. Xu, Y. S. Rawat, Y. Wong, M. Kankanhalli, and M. Shah, "Don't pour cereal into coffee: Differentiable temporal logic for temporal action segmentation," in *Adv. Neural Inf. Process. Syst.*, 2022.

[16] 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.

[17] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, 2005, pp. 729–734.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[19] S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi, "Symbolic LTLf synthesis," in *Int. Joint Conf. Artif. Intell.*, 2017, pp. 1362–1369.

[20] A. Camacho, J. Baier, C. Muise, and S. McIlraith, "Finite LTL synthesis as planning," in *Proc. Int. Conf. Automated Planning and Scheduling*, vol. 28, 2018, pp. 29–38.

[21] Y. Xie, F. Zhou, and H. Soh, "Embedding symbolic temporal knowledge into deep sequential models," in *IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4267–4273.

[22] P. Vaezipoor, A. C. Li, R. A. T. Icarte, and S. A. Mcilraith, "LTL2Action: Generalizing LTL instructions for multi-task RL," in *Int. Conf. Mach. Learn.*, 2021, pp. 10 497–10 508.

[23] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, 2018, pp. 593–607.

[24] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artif. Intell.*, vol. 116, no. 1, pp. 123–191, 2000.

[25] J. A. Baier and S. A. McIlraith, "Planning with temporally extended goals using heuristic search," in *Proc. Int. Conf. Automated Planning and Scheduling*, 2006, p. 342–345.

[26] A. Camacho, E. Triantafillou, C. Muise, J. Baier, and S. McIlraith, "Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017.

[27] G. Fainekos, H. Kress-Gazit, and G. Pappas, "Temporal logic motion planning for mobile robots," in *IEEE Int. Conf. Robot. Automat.*, 2005, pp. 2020–2025.

[28] V. Kurtz and H. Lin, "Temporal logic motion planning with convex optimization via graphs of convex sets," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3791–3804, 2023.

[29] C. Yang, M. L. Littman, and M. Carbin, "On the (in)tractability of reinforcement learning for LTL objectives," in *Int. Joint Conf. Artif. Intell.*, 2022, pp. 3650–3658.

[30] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an RL agent using LTL," in *Proc. Int. Conf. Autonomous Agents Multiagent Syst.*, 2018, pp. 452–461.

[31] C. Voloshin, H. M. Le, S. Chaudhuri, and Y. Yue, "Policy optimization with linear temporal logic constraints," in *Adv. Neural Inf. Process. Syst.*, 2022, pp. 17 690–17 702.

[32] J. Song, Q. Zhang, H. Yin, M. Mardani, M.-Y. Liu, J. Kautz, Y. Chen, and A. Vahdat, "Loss-guided diffusion models for plug-and-play controllable generation," in *Int. Conf. Mach. Learn.*, vol. 202, 2023, pp. 32 483–32 498.

[33] P. Dhariwal and A. Q. Nichol, "Diffusion models beat GANs on image synthesis," in *Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 8780–8794.

[34] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[35] A. Jalal, M. Arvinte, G. Daras, E. Price, A. G. Dimakis, and J. Tamir, "Robust compressed sensing MRI with deep generative priors," in *Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 14 938–14 954.

[36] H. Chung and J. C. Ye, "Score-based diffusion models for accelerated MRI," *Med. Image Anal.*, p. 102479, 2022.

[37] H. Chung, B. Sim, and J. C. Ye, "Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12 413–12 422.

[38] H. Chung, B. Sim, D. Ryu, and J. C. Ye, "Improving diffusion models for inverse problems using manifold constraints," in *Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 25 683–25 696.

[39] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142–153, 2017.

[40] K. Chen, E. Lim, K. Lin, Y. Chen, and H. Soh, "Behavioral refinement via interpolant-based policy diffusion," in *Proc. Robot.: Sci. and Syst.*, 2024.