This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

Actively Learning Task Constraints from Feasible and Failed Demonstrations

Baiyu Peng[1], Aude Billard[1]

***Abstract*—In many autonomous control and robotic tasks, agents must operate under constraints; however, many of these constraints are unknown or difficult to specify explicitly. This paper addresses the problem of learning unknown constraints from demonstrations, including not only feasible demonstrations but also failed ones, which are typically ignored by existing methods. We propose Multiple Instance Constraint Learning (MICL), a framework that treats feasible and failed demonstrations as bags of states with different bag-level feasibility labels, and infers a state-level constraint function using a constraint network with a dedicated pooling layer. Theoretical analysis shows that MICL can identify infeasible states with bounded errors, and thus recover the underlying constraint function. Unlike existing works that learn only from static datasets, we further introduce an active learning framework that enables the agent to iteratively refine the constraint model by querying an expert about the feasibility of newly generated trajectories. These trajectories are generated using either a Dynamical System or a Reinforcement Learning policy, and selected for querying based on uncertainty-, diversity-, or mutual-information-based scores. We validate the proposed method in five simulated robotic tasks with different state dimensions and constraint structures. Results show that MICL outperforms or matches the strongest baselines in terms of constraint accuracy and downstream policy violation rate. The active learning framework progressively improves constraint accuracy as more expert-labeled queries are added, and active query strategies generally outperform random querying by selecting more informative trajectories.**

***Impact Statement*—Constraints play a fundamental role in many autonomous-control settings, including robotics, where safety, feasibility, and physical limitations must be respected. However, many constraints are unknown in advance or hard to specify explicitly, giving rise to the problem of constraint learning. A common formulation of constraint learning assumes a known reward function and learns only from feasible demonstrations collected in fixed datasets, where failures are often excluded and learning remains passive. This article contributes two complementary perspectives. First, it shows that failed demonstrations can be systematically leveraged, together with feasible ones, to provide informative contrast for identifying constraints. Second, it studies how constraint learning can move beyond static datasets by enabling agents to actively query experts and refine constraint models based on informative feedback. Together, these perspectives broaden the sources of supervision and data-acquisition mechanisms considered in constraint learning, suggesting a route toward more adaptive and data-efficient constraint learning for autonomous-control systems.**

***Index Terms*—Active learning, constraint inference, learning from demonstration, safe motion planning**

[1] The authors are with the LASA, School of Engineering, EPFL (Swiss Federal Institute of Technology in Lausanne), Lausanne 1015 Vaud, Switzerland. (baiyu.peng@epfl.ch ; aude.billard@epfl.ch). Corresponding author: Baiyu Peng.

This work was supported by euROBIN Grant. All code and datasets are available at https://github.com/epfl-lasa/Actively_Learning_Constraints_from_Failed_Demonstrations

I. Introduction

SAFE operation of autonomous agents often relies on the explicit identification and enforcement of constraints, which specify states or actions that should be avoided to satisfy safety, feasibility, or physical requirements [1]–[3]. In many scenarios, however, such constraints may be unknown or difficult to specify analytically, especially when they are nonlinear, have unknown boundary shapes, or depend on task-specific preferences. For example, in a robot manipulation task, a user may implicitly expect the robot to keep different safety margins around fragile or irregular objects, but these margins may not be available as explicit mathematical constraints. Alternatively, it is possible to obtain demonstrations from the users and discover the underlying constraints from these demonstrations, as they inherently encode critical safety requirements and human preferences [4]–[6].

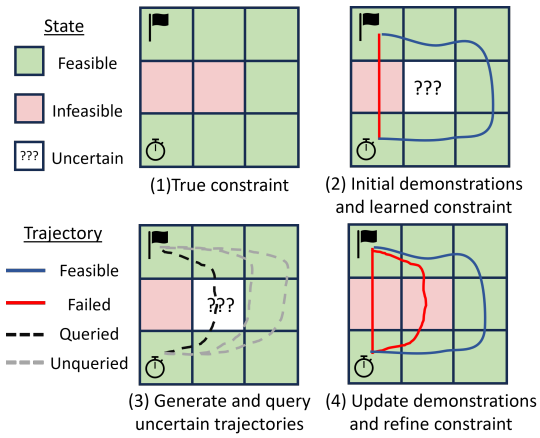


Fig. 1. A toy example of active constraint learning from feasible and failed demonstrations in a 3×3 grid world. The agent starts from the lower-left cell and aims to reach the upper-left cell. Green, red, and white cells denote feasible, infeasible, and uncertain states, respectively. Blue and red curves denote feasible and failed demonstrations, while dashed curves denote candidate query trajectories. (1) The true constraint is to avoid the two infeasible cells shown in red. (2) Initially, one feasible and one failed demonstration are provided. From these demonstrations, the left-middle cell can be inferred as infeasible, but the middle cell remains uncertain because it has not been visited. (3) The agent generates candidate trajectories and queries the expert about the feasibility of the most informative one, shown in black. (4) The queried trajectory is labeled failed and added to the demonstration set, allowing the learned constraint model to be refined.

Constraint inference from demonstrations has attracted increasing attention recently. While most early works in the field only learn specific types of constraints (e.g., move-in-line constraint [7] or fixed-point constraint [8]) from a predefined constraint-prototype library, recent works focus on

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

learning a more general and flexible constraint function of the states or state-action pairs [5], [9]. In this work, we focus on learning constraints that explicitly require the agent to avoid infeasible states (or state-action pairs) during a trajectory. These infeasible states correspond to conditions or behaviors that the user aims to prevent, such as collisions with unknown obstacles or exceeding unknown velocity limits, which are prevalent in real-world applications.

Most existing methods for learning such constraints mainly follow inverse constrained reinforcement learning (ICRL) or inverse optimal control (IOC) frameworks. ICRL approaches, such as maximum entropy constraint learning [5], [10], [11], are directly inspired by and extended from inverse reinforcement learning (IRL). In contrast, IOC methods rely more on the closed-form dynamics model and the KKT condition [9], [12], [13]. Despite their differences, both paradigms usually assume a known task reward and near-optimal demonstrations. They infer constraints by comparing demonstrated behavior with reward-maximizing alternatives. Regions frequently visited by an unconstrained reward-maximizing policy, but rarely visited by the demonstrations, are treated as candidates for infeasibility. In ICRL, this idea is often implemented through a maximum-entropy framework [5], whereas IOC methods use tools such as mixed-integer programming [4] and KKT conditions [12].

Despite the success of existing constraint learning methods, most of them rely primarily on feasible demonstrations that rarely or never violate the constraints. In practice, however, data collection may naturally produce many failed attempts caused by violations of unknown constraints [14]. In this paper, we use the term failed demonstrations specifically to denote trajectories that violate unknown constraints, although failures may have other meanings in different contexts. Such failed demonstrations are usually discarded in constraint learning, but they can provide valuable information because they visit infeasible states that caused the failure. Leveraging them can therefore help identify infeasible regions more directly than relying on feasible demonstrations alone. In tasks where violations are not catastrophic, one may even deliberately collect these failed data to indicate undesirable behavior.

In addition to that, most existing works are trained on a fixed set of demonstrations and lack mechanisms for collecting new data and continuously refining the learned constraint models. In some tasks, however, although the demonstrator may not be able to mathematically specify the constraints, they are often capable of evaluating the feasibility of newly generated trajectories. In such scenarios, an active framework that incrementally improves the constraint model based on user feedback is preferable. This active learning process is particularly valuable when learning from failed demonstrations, since these demonstrations might not cover the entire state space.

Fig. 1 illustrates the motivation for learning from failed demonstrations and active constraint learning. Initial feasible and failed demonstrations may cover only part of the state space, leaving some regions uncertain. Instead of relying on a fixed dataset, the agent can generate candidate trajectories, query the expert for trajectory-level feasibility labels, and iteratively refine the learned constraint model with the newly labeled data.

Motivated by these observations, this paper introduces a novel active constraint learning method to uncover constraints from both feasible and failed demonstrations. Specifically,

1) We propose Multiple Instance Constraint Learning (MICL), a framework that can learn constraints from both feasible and failed demonstrations. It treats both types of demonstrations as bags of states, where the bag feasibility labels are known but state feasibility labels remain unknown, and extracts the infeasible states by fitting a constraint network with a multiple-instance pooling layer.

2) We provide a theoretical analysis under mild assumptions showing that the proposed MICL can provably identify infeasible states with bounded errors and thus recover the underlying constraint function.

3) We introduce an active constraint learning framework that allows the agent to iteratively refine the constraint model by synthesizing new trajectories and querying an expert about its feasibility. The query is decided based on an uncertainty-diversity score function.

4) We provide simulation validation across five constraint learning tasks with different dimensions and constraint structures.

II. Related Works

We review the existing constraint learning methods via the inverse constrained reinforcement learning (ICRL) approach and the inverse optimal control (IOC) approach, respectively.

The initial ICRL framework was proposed by [15], who modeled the constraint learning problem through maximum likelihood inference. In their formulation, feasible trajectories are assigned a likelihood proportional to the exponential returns, whereas failed trajectories are given zero likelihood. They further introduced a greedy algorithm designed to incrementally add the minimal number of infeasible states required to maximize the demonstration likelihood. Building upon this foundation, [16] and [17] extended the maximum likelihood approach to stochastic environments by incorporating soft or probabilistic constraints. [10] proposes to learn a neural network policy via deep RL and to identify discrete infeasible states with the policy network. Nevertheless, the aforementioned methods all rely on a predefined finite constraint set. To overcome this limitation and enable learning of a more flexible constraint function, [5] introduced the Maximum Entropy Constraint Learning (MECL) framework, which jointly trains a policy network and a constraint network. The constraint network is optimized by performing gradient ascent on the maximum likelihood objective. Later, [11] proposes a variational approach to infer the distribution of the constraint, which captures its epistemic uncertainty. This is followed by an improved work that simultaneously handles both aleatoric and epistemic uncertainty via flow-based generative data augmentation [18]. More recently, researchers have also explored learning with offline data [19], confidence-aware objectives [20], and soft constraints [21].

IOC approaches are initially built on the intuition that any trajectory yielding a higher reward than the demonstrated ones

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

must be restricted by some unknown constraint; otherwise, the demonstrator could have followed it. In [4], the authors employ hit-and-run sampling to generate such higher-reward trajectories and formulate an integer programming problem to identify infeasible states in a grid world setting. This framework was later extended to continuous state spaces, where constraints are modeled as combinations of axis-aligned boxes through unions and intersections [12]. While the previous two methods are model-free, the same group further proposed IOC methods that exploit known environmental models and leverage the KKT optimality conditions to recover either parametric constraint representations [12] or non-parametric constraints modeled via Gaussian Processes [9].

Despite the success of the aforementioned works, none of them explicitly learns constraints from failed demonstrations. Although some methods that learn soft constraints are compatible with partially failed demonstrations [16], [21], they only treat these demonstrations as noisy data that are processed in a robust manner.

Besides, most of the existing methods assume that training takes place on a pre-collected, fixed demonstration set and offer no guidance for acquiring new data to refine the constraint model. The two exceptions are [20] and [22]. However, [20] only defines a metric to decide whether the current demonstration set is sufficient for training a high-confidence model, without addressing how to collect the new demonstrations. [22] is the only work that explores active constraint learning. However, it queries the expert on the feasibility of individual state-action pairs, whereas our work queries the expert on the feasibility of entire trajectories, which is more practical but more challenging.

Finally, active preference-based reward learning is also related to our work in its use of human feedback and active query selection. These methods typically query pairwise preferences between two trajectories or trajectory segments and use the feedback to infer an underlying reward function [23], [24]. Some works further study batch query generation or uncertainty-based exploration to improve feedback efficiency [25], [26]. However, the feedback format is also different: preference-based reward learning usually queries relative preferences between two trajectory segments, while our framework queries a binary feasible/failed label for a single trajectory and uses such trajectory-level labels to recover state-level constraints.

III. Problem Formulation

Preliminary. We consider a finite-horizon deterministic Markov decision process (MDP) [27], where the states and actions are denoted by $s \in \mathcal{S}$ and $a \in \mathcal{A}$. γ denotes the discount factor, and the real-valued reward is denoted by $r(s,a)$. A trajectory $\tau = \{(s_1,a_1),\ldots,(s_T,a_T)\}$ contains a sequence of state-action pairs in one episode. For the trajectory τ, the total discounted reward (return) is defined as $r(\tau) = \sum_{t=1}^{T} \gamma^t r(s_t,a_t)$. A policy, the mapping between states and actions, is denoted by $a = \pi(s)$.

In this study, we focus on learning Markovian state–action constraints that prohibit the agent from visiting infeasible states or actions throughout a trajectory. Mathematically, we denote the true constraint set by $\mathcal{C}^* = \{(s,a) \in \mathcal{S} \times \mathcal{A} | (s,a) \text{is truly infeasible}\}$. Thus, the true constraint is written as $(s,a) \notin \mathcal{C}^*$. This definition implicitly assumes that the studied constraint is time-independent and deterministic. In addition to the unknown constraints, some known constraints could also be present in the task, e.g., goal state constraints and system dynamics constraints. For simplicity of explanation, we assume all demonstrations satisfy the known constraints, since those that violate the known constraints can be discarded.

To simplify the notation and allow for more flexibility, we introduce a feature state $\bar{s} \equiv \psi(s,a)$ as the input to the constraint function, where $\psi(s,a)$ is a user-defined feature function that can be customized to incorporate prior knowledge about the constraint structure. For example, if the constraint is known to depend solely on the state (or actions), one could define $\psi(s,a) = s$ (or a). Readers should keep in mind that, in the following sections, we use the feature state $\bar{s}$ as shorthand for $\psi(s,a)$ and often simply refer to it as the state.

Correspondingly, we denote the ground-truth feasibility of $\bar{s}$ as $y(\bar{s}) \in \{0,1\}$ (1 = feasible). Here, feasibility refers to aggregate feasibility with respect to all unknown constraints: if multiple semantic constraints are present, $\bar{s}$ is feasible only when all of them are satisfied, and is infeasible if any one of them is violated. We approximate the true feasibility with a constraint network $\zeta_\theta(\bar{s}) \in [0,1]$, parameterized by θ. $\zeta_\theta(\bar{s})$ induces a binary feasibility prediction $\hat{y}(\bar{s}) = \mathbf{1}[\zeta_\theta(\bar{s}) \geq d]$ and a learned constraint set $\mathcal{C}_\theta = \{(s,a)|\hat{y}(\bar{s}) = 0\}$, with $d = 0.5$ as the decision threshold.

Demonstrations and Expert Availability. Our method assumes access to both a feasible demonstration set and a failed demonstration set, which are obtained and labeled by a human expert before training. The feasible demonstration set $\mathcal{D} = \{\tau_i\}_{i=1}^{N_d}$ comprises feasible trajectories that always satisfy the unknown constraints, i.e., $\forall (s,a) \in \tau \in \mathcal{D}, (s,a) \notin \mathcal{C}^*$. Likewise, the failed demonstration set $\mathcal{F} = \{\tau_i\}_{i=1}^{N_f}$ comprises failed trajectories where each trajectory has violated the constraint at least once, i.e., $\forall \tau \in \mathcal{F}, \exists (s,a) \in \tau, (s,a) \in \mathcal{C}^*$.

Our method does not assume that the demonstrations are optimal. Moreover, for non-active learning settings, our approach does not rely on access to a known reward function. These two aspects constitute key advantages of our method. However, in practice, if the number of demonstrations is limited and the covered state space is small, we expect both feasible and failed demonstrations to aim at accomplishing the same task, with their distinction lying primarily in whether the constraints are violated.

To enable active learning, we require that an expert is available during learning, allowing the agent to repeatedly present uncertain trajectories τ_q to the expert for feasibility labeling. The labeled trajectories will be incorporated into the feasible/failed demonstration set accordingly and utilized in subsequent training. It is worth noting that a related work [22] also studies active constraint learning, but proposes querying the feasibility of individual state-action pairs, instead of a whole trajectory. In contrast, we argue that it is more natural to ask the expert to evaluate a whole trajectory, since most real-world tasks are accomplished and evaluated at a trajectory

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

level. Compared with a state-level label, a trajectory-level label is a much weaker supervision and thus presents a greater challenge for learning.

IV. Learning Constraints from Feasible and Failed Demonstrations

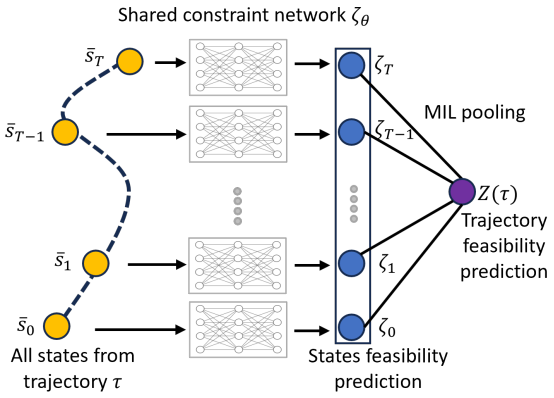


Fig. 2. The forward prediction of the proposed multiple instance constraint learning method. The same constraint network takes each state $\bar{s}$ from a given trajectory τ as input, and outputs its corresponding state feasibility prediction $\zeta(\bar{s})$. These state-level predictions are then aggregated using a MIL pooling layer to produce a final trajectory-level feasibility prediction $Z(\tau)$.

A. Method: Multiple Instance Constraint Learning

We aim to extract infeasible states from the feasible demonstrations $\mathcal{D}$ and failed demonstrations $\mathcal{F}$. Although we are certain that any failed demonstration must have visited some infeasible states, it remains unclear which specific states within the trajectory are infeasible. In contrast, all states from feasible demonstrations are guaranteed to be truly feasible. Our objective is to train a constraint function using these two datasets to classify each state as either feasible or infeasible. However, the available trajectory-level labels cannot be directly utilized by standard classification tools.

To tackle this problem, we draw inspiration from Multiple Instance Learning (MIL) and propose a Multiple Instance Constraint Learning (MICL) method. MIL is a machine learning paradigm, where training instances are arranged in sets, usually called bags, and only bag-level labels are available during training while instance-level labels are absent [28], [29]. The MIL setting closely resembles constraint learning, where we only have a trajectory-level label (feasible/failed) of demonstrations but lack state-level labels of each state from the demonstrations. We aim to train a state-level constraint function given only trajectory-level labels.

To achieve this, our MICL creates a networked constraint model composed of a fully connected network (FNN) and an MIL pooling layer, as illustrated in Fig. 2. The FNN, parameterized with θ, takes each state $\bar{s}$ from a trajectory τ as input, and outputs its corresponding feasible probability $\zeta_\theta(\bar{s})$. During training, since only trajectory-level labels are available, we treat the state feasible probability $\zeta(\bar{s})$ as a latent variable and introduce an additional MIL pooling layer to aggregate all $\zeta(\bar{s})$ into a trajectory-level feasibility prediction, denoted as $Z(\tau)$. The true feasibility labels of feasible demonstrations are set to 1, and those of failed demonstrations are 0. The network is trained by minimizing the binary cross-entropy loss between the prediction $Z(\tau)$ and the corresponding ground-truth labels. The gradient of the loss is shown in (1). During inference, one can obtain the feasible probability of a given state $\bar{s}$ solely via FNN, omitting the pooling layer. The threshold for the state being feasible is always set to be 0.5.

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta) = -\mathbb{E}_{\tau\sim\mathcal{D}\cup\mathcal{F}}[\mathbb{1}\left[\tau\in\mathcal{D}\right]\nabla_\theta \log Z(\tau) \\ +\mathbb{1}\left[\tau\in\mathcal{F}\right]\nabla\theta\log(1-Z(\tau))].\end{aligned} \tag{1}$$

The design of the MIL pooling layer plays a critical role in constraint learning. In this work, we adopt the Log-sum-exp (LSE) pooling, which offers a flexible and smoother training process.

$$Z(\tau) = 1 - \omega \log\left[\frac{1}{|\tau|}\sum_{(s,a)\in\tau}\exp\left(\frac{1-\zeta_\theta(\bar{s})}{\omega}\right)\right] \tag{2}$$

where ω in LSE is a parameter controlling the smoothness of the function. The MIL pooling layer acts as a (soft) minimum layer. It aligns with our constraint learning setting: if the feasible probability of at least one state s from a trajectory τ is low, the feasible probability of that trajectory should be low. The tunable parameter ω actually plays the role of regularization and enables the user to control the tightness of the constraint network. A small ω approximates the minimization better, and thus a failed demonstration only needs to have one infeasible state. This encourages the network to predict fewer infeasible states. Based on our tuning experience , setting ω within the range of 0.05 to 0.5 yields decent performance across different tasks. A sensitivity analysis on ω is later provided in section VI-A.

The above formulation can also handle tasks with multiple unknown constraints. In such cases, $\zeta_\theta(\bar{s})$ should be interpreted as an aggregate feasibility function: a state is feasible only if all unknown constraints are satisfied, and becomes infeasible once any one of them is violated.

The MICL framework thus provides a unified way to extract state-level constraints from feasible and failed demonstrations. Despite its simple yet principled formulation, it is important to theoretically justify why MICL is indeed effective for constraint learning. Therefore, in the following subsection, we establish the learnability of MICL under assumptions tailored to the constraint learning setting, showing that the proposed framework can recover the underlying constraint with bounded errors.

B. Learnability Analysis

There is a substantial body of theory analyzing learnability in the multiple instance learning setting. Classical results include VC-dimension analyses [30], latent model formulations [31], and margin-based bounds [30]. However, most of these works study only bag-level learnability, i.e., whether one can correctly predict the label of an entire bag (corresponding to a trajectory) after training.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

In contrast, the constraint learning problem requires not only accurate prediction of a trajectory's feasibility but also the identification of the feasibility of each state on that trajectory. A handful of works on MIL [31], [32] have analyzed instance-level (state-level) learnability, but these analyzes typically assume that instances within the same bag are drawn i.i.d. from a single underlying distribution. This assumption may be valid in some domains but fails for constraint learning, where each bag (trajectory) is a temporally correlated sequence of states rather than an i.i.d. sample set. In our case, if one trajectory has visited one infeasible state, it is more likely to visit more in the future, since the infeasible states are usually concentrated and close to each other.

Building on these insights, we first establish the trajectory-level learnability using VC-dimension theories and then derive a novel form of state-level learnability under assumptions that reflect the temporal and structural nature of constraint learning.

Trajectory-level Learnability. Let $\mathcal{H}_{\text{state}}$ denote the hypothesis class of state-level constraint function $\hat{y} : \bar{s} \mapsto \{0, 1\}$, with VC dimension d_v. In practice, $\hat{y}(\bar{s})$ is output by our neural network constraint model ζ with $\hat{y}(\bar{s}) = \mathbf{1}[\zeta(\bar{s}) \geq 0.5]$. Let $\mathcal{H}_{\text{traj}}$ denote the hypothesis class of trajectory-level constraint functions $\hat{y} : \tau \mapsto \{0, 1\}$, where $\hat{y}(\tau)$ is induced by aggregating state-level outputs through a pooling function Z:

$$Z(\tau) = Z\big(\zeta(s_1), \ldots, \zeta(s_T)\big), \quad T \leq T_{\max}, \tag{3}$$

$$\hat{y}(\tau) = \mathbf{1}[Z(\tau) \geq 0.5] \tag{4}$$

where Z is one of the pooling functions defined in (2), $T_{\max}$ is the maximum trajectory length in the dataset and $\hat{y}(\tau)$ is the feasibility prediction. Denoting $y(\tau) \in \{0, 1\}$ as the ground-truth feasibility label of trajectory τ, the trajectory-level error of the predictor $\hat{y}(\tau)$ is defined as

$$\mathcal{E}_{\text{traj}}(Z) = \Pr_{\tau \sim \Gamma}\big[\hat{y}(\tau) \neq y(\tau)\big], \tag{5}$$

where Γ is the underlying distribution from which both successful and failed demonstrations are drawn. Correspondingly, the empirical risk is defined on the given demonstration sets as

$$\hat{\mathcal{E}}_{\text{traj}}(Z) = \frac{1}{|\mathcal{D}| + |\mathcal{F}|} \sum_{\tau \in \mathcal{D} \cup \mathcal{F}} \big[\hat{y}(\tau) \neq y(\tau)\big], \tag{6}$$

Lemma 1 (Aggregated VC-dimension [30])**.** *If the state-level hypothesis class $\mathcal{H}_{state}$ has VC dimension d_v, then for any monotone pooling function Z aggregating up to $T_{\max}$ states, the induced trajectory-level class satisfies*

$$\mathrm{VC}(\mathcal{H}_{traj}) = O(d_v \log T_{\max}). \tag{7}$$

Remarks. This lemma is a direct application of MIL results [30] into the constraint learning context. Intuitively, although trajectories can contain up to $T_{\max}$ states, the complexity only grows logarithmically with $T_{\max}$ relative to the state-level class.

Corollary 1 (Trajectory-level learnability)**.** *For any $\delta \in (0, 1)$ and n independent trajectories, the empirical risk minimizer $\hat{Z} = \arg\min_{Z \in \mathcal{H}_{traj}} \hat{\mathcal{E}}_{traj}(Z)$ satisfies, with probability at least $1 - \delta$,*

$$\mathcal{E}_{traj}(\hat{Z}) \leq \inf_{Z \in \mathcal{H}_{traj}} \mathcal{E}_{traj}(Z) + O\left(\sqrt{\frac{d_v \log T_{\max} + \log(1/\delta)}{n}}\right). \tag{8}$$

Remarks. This bound is obtained by taking 1 into the standard VC generalization model [33] assuming exact empirical risk minimization with respect to the 0–1 loss. In practice, our algorithm minimizes a surrogate loss (cross-entropy) via gradient-based optimization. Since cross-entropy is classification-calibrated [34], minimizing surrogate risk guarantees consistency with respect to the 0-1 loss. Therefore, our MICL algorithm inherits the learnability guarantee of the hypothesis class, up to an additional optimization error term that accounts for approximation and optimization gaps. This establishes that our constraint learning method is PAC-learnable at the trajectory level, with sample complexity depending logarithmically on $T_{\max}$.

State-level Learnability. We now establish state-level learnability by relating trajectory-level errors to state-level errors. Two types of state-level errors are discussed separately: *false feasible* (FF, a truly infeasible state predicted feasible) and *false infeasible* (FI, a truly feasible state predicted infeasible).

1) FF error: The expected state-level false feasible (FF) error is formally defined as

$$\mathcal{E}^{\text{FF}}_{\text{state}} = \Pr\big[\hat{y}(\bar{s}) = 1 \mid y(\bar{s}) = 0\big], \tag{9}$$

where $\hat{y}(\bar{s})$ is the state prediction (1 = feasible), and $y(\bar{s})$ is the ground-truth feasibility. We also introduce the expected trajectory-level FF rate:

$$\mathcal{E}^{\text{FF}}_{\text{traj}} = \Pr\big[\hat{y}(\tau) = 1 \mid y(\tau) = 0\big]. \tag{10}$$

For any truly infeasible trajectory τ, denote

$$K_\tau = \{\bar{s} \in \tau \mid y(\bar{s}) = 0\} \quad \text{and} \quad M = \max_{\tau} |K_\tau|. \tag{11}$$

That is, K_τ is the set of all infeasible states in τ and M is a maximum number of infeasible states that appear in any infeasible trajectory drawn from Γ. In practice, $|K_\tau|$ is typically much smaller than the trajectory length, but we use the upper bound M for analytical convenience. For an infeasible trajectory τ, define per-state FF rate $q_{\bar{s}_t, \tau}$

$$q_{\bar{s}_t, \tau} = \Pr\big[\hat{y}(\bar{s}_t) = 1 \mid \tau\big], \quad \forall \bar{s}_t \in K_\tau, \tag{12}$$

and the average FF rate $\bar{q}_\tau$

$$\bar{q}_\tau = \frac{1}{|K_\tau|} \sum_{\bar{s}_t \in K_\tau} q_{\bar{s}_t, \tau} \tag{13}$$

Assumption 1 (Bounded heterogeneity of per-state FF errors)**.** *For each truly infeasible trajectory τ, there exists $\rho \in (0, 1]$ such that*

$$\min_{\bar{s} \in K_\tau} q_{\bar{s}, \tau} \geq \rho\, \bar{q}_\tau. \tag{14}$$

Remarks. This assumption indicates that the FF error rate of any infeasible state within a trajectory cannot be significantly lower than the average trajectory FF rate. Intuitively, this means the state FF error can happen everywhere, rather than only on a small subset of states. It is notable that the strength of this effect also relies on ρ, a task-specific coefficient. Larger values of ρ indicate stronger assumption and will yield tighter bounds, while smaller values result in looser bounds.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

Assumption 2 (Positive correlation of FF errors)**.** *For each truly infeasible trajectory τ, there exists $\underline{\alpha} \geq 1$ such that*

$$\Pr\Big(\bigcap_{\bar{s}\in K_\tau} \{\hat{y}(\bar{s}) = 1\} \mid \tau \Big) \geq \underline{\alpha} \prod_{\bar{s}\in K_\tau} q_{\bar{s},\tau}. \tag{15}$$

Remarks. This assumption indicates that FF errors tend to co-occur, i.e., the probability that all infeasible states are simultaneously misclassified (left-hand side) is larger than the product of the individual FF rate (right-hand side). This is reasonable in constraint learning because infeasible states are often spatially concentrated—for example, all states inside an obstacle—so their classification errors tend to co-occur. The strength of this correlation is quantified by a task-specific coefficient $\underline{\alpha} \geq 1$: larger values correspond to a stronger co-occurrence of FF errors and will yield tighter bounds, while lower values yield looser bounds. The special case $\underline{\alpha} = 1$ recovers independence between state-level errors.

Theorem 1 (State-level learnability)**.** *Under Assumptions 1 and 2 , the trajectory-level false-feasible rate $\mathcal{E}^{FF}_{traj}$ is lower-bounded by the state-level false-feasible rate $\mathcal{E}^{FF}_{state}$ as*

$$\mathcal{E}^{FF}_{state} \leq \left(\frac{\mathcal{E}^{FF}_{traj}}{\underline{\alpha}\, \rho^M} \right)^{1/M}. \tag{16}$$

Proof: See Appendix A. ∎

Remarks 1. This theorem establishes a quantitative link between trajectory-level and state-level FF errors. It shows that the state-level FF rate decays polynomially with the trajectory-level rate, with exponent $1/M$ governed by the maximum number of infeasible states per trajectory. A larger ρ (more uniform per-state errors) and a larger $\underline{\alpha}$ (stronger positive correlation) both lead to a tighter state-level guarantee.

Remarks 2. Taking (16) into the trajectory-level error bound, we will immediately have an error bound for the state-level FF error bound. But note that the trajectory-level FF rate $\mathcal{E}^{\text{FF}}_{\text{traj}}$ is defined over infeasible trajectories, whereas $\mathcal{E}_{\text{traj}}$ in (8) is averaged over all trajectories. Although the PAC bound in (8) formally applies to the full distribution, the conditional subset of infeasible trajectories follows the same hypothesis class $\mathcal{H}_{\text{traj}}$ and thus exhibits the same asymptotic generalization behavior, up to the sampling proportion of infeasible trajectories.

2) FI error: Unfortunately, unlike the FF error, the state-level FI rate $\mathcal{E}^{\text{FI}}_{\text{state}} = \Pr[\hat{y}(\bar{s}) = 0 \mid y(\bar{s}) = 1]$ cannot be bounded by the trajectory-level error. To see this, one can consider an extreme case. Even if the trajectory classifier achieves perfect accuracy (i.e., $\mathcal{E}_{\text{traj}} = 0$), the state-level FI error may still be large: a truly infeasible trajectory may contain many feasible states that are misclassified as infeasible, which does not cause trajectory-level error but increases the state-level FI rate. Therefore, state-level FI error bound cannot be obtained without additional strong structural assumptions.

Discussions. The above analysis establishes PAC-learnability at the state level for false feasible errors but not for false infeasible ones. This limitation is, however, acceptable in the context of constraint learning: the primary goal of the constraint model is to identify infeasible regions reliably. As long as infeasible states are learnable in the PAC sense, the resulting policy is less likely to violate true constraints, although it might become overly conservative due to unbounded FI errors. As for the trajectory-level learnability, our method guarantees the feasibility is learnable for both FF and FI errors.

V. Active Refinement of Constraints

Building upon the MICL formulation, we now introduce an active constraint learning framework in which the agent can iteratively refine the constraint model by querying the expert about the feasibility of selected trajectories.

A. Framework Overview

The overall framework of the proposed active constraint learning process is illustrated in Fig. 3. After initial training with a given set of feasible and failed demonstrations, the agent proceeds as follows in each iteration: it first generates a candidate trajectory set $\mathcal{T}$ by rolling out trajectories from random starting points. Then, based on a query score function $Q(\tau)$, which evaluates the informativeness of each trajectory (e.g., using uncertainty and diversity), the agent selects the top N_q trajectories for expert labeling. Each queried trajectory is presented to the expert [1], labeled as feasible or infeasible, and appended to the corresponding demonstration set $\mathcal{D}$ or $\mathcal{F}$. The constraint network is subsequently updated with the expanded dataset. This process repeats until a stopping condition is met, such as reaching a maximum number of iterations or when the user deems recent query results satisfactory. The pseudocode of the proposed algorithm is attached below. A computational complexity analysis is also attached in the Appendix C.

Algorithm 1 Active Constraint Learning From Feasible and Failed Demonstrations

1: **input:** Feasible demonstration set $\mathcal{D}$, failed demonstration set $\mathcal{F}$, randomly initialized constraint network ζ_θ
2: Train the constraint network ζ_θ with two initial demonstration sets
3: **repeat**
4: Update the policy π based on the current constraint network ζ_θ, using either the modulated DS or RL
5: Sample N_s candidate trajectories using the updated policy π and compute their query scores
6: Select the top N_q trajectories with the highest query scores and query the expert for their feasibility
7: Append the newly labeled trajectories to the demonstration set $\mathcal{D}$ or $\mathcal{F}$ according to the expert labels
8: Update the constraint ζ_θ with the updated demonstration sets
9: **until** reach maximum iterations

In the following two subsections, we first elaborate on the query strategy for picking queries from the candidates $\mathcal{T}$ and discuss the candidate trajectory generation approach.

[1]If constraint violations do not lead to dangers, the trajectory can be executed and played directly on the real-world agent; otherwise, it is executed in simulation.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

B. Query Scoring and Selection

A common active-learning strategy is to query samples that are most informative under the current model [35]. In our setting, each query is a trajectory and the expert provides only a trajectory-level feasible/failed label. We therefore define a trajectory-level informativeness score $U(\tau)$, and compare three query strategies.

1) **Temperature-Scaled Entropy** [36]: This strategy measures predictive uncertainty through the entropy after temperature scaling. For a state $\bar{s}$, the scaled feasible probability is

$$\zeta_T(\bar{s}) = \frac{\exp(z_1(\bar{s})/T)}{\exp(z_0(\bar{s})/T) + \exp(z_1(\bar{s})/T)}, \tag{17}$$

where $z_i(\bar{s})$ is the logit of class i, and T is the temperature. The state-level uncertainty is the binary entropy of $\zeta_T(\bar{s})$, and $U(\tau)$ is obtained by averaging it over the trajectory.

2) **Monte Carlo Dropout** [37]: This strategy treats a dropout network as an approximate Bayesian model. During query selection, dropout is kept active and M stochastic forward passes are performed. Let θ_m denote the effective network parameters induced by the m-th dropout mask, and let $\zeta_{\theta_m}(\bar{s})$ be the corresponding state-level feasible probability. The strategy denoted as Dropout scores a trajectory by the average predictive variance across dropout samples:

$$U_{\text{Dropout}}(\tau) = \frac{1}{|\tau|} \sum_{(s,a)\in\tau} \text{Var}_{m=1}^{M}[\zeta_{\theta_m}(\bar{s})]\,. \tag{18}$$

3) **Bayesian Active Learning by Disagreement (BALD)** [38]: BALD follows the same MC-dropout stochastic predictions, but replaces predictive variance with a mutual-information score. For a candidate trajectory τ, let $p_m(\tau)$ be the trajectory-level feasible probability obtained from the m-th stochastic dropout forward pass after applying the MICL pooling layer to $\{\zeta_{\theta_m}(\bar{s})\}_{(s,a)\in\tau}$. We define the mean predictive probability as $\bar{p}(\tau) = \frac{1}{M}\sum_{m=1}^{M} p_m(\tau)$. The BALD score is

$$U_{\text{BALD}}(\tau) = H(\bar{p}(\tau)) - \frac{1}{M}\sum_{m=1}^{M} H(p_m(\tau)), \tag{19}$$

where $H(p) = -p\log p - (1-p)\log(1-p)$ is the binary entropy. This score favors trajectories on which different dropout posterior samples disagree.

We also include a diversity score to encourage exploration of data-sparse regions. For a state $\bar{s}$, the diversity score is

$$d(\bar{s}) = \frac{1}{k} \sum_{\bar{s}'\in kNN(\bar{s},\mathcal{D}\cup\mathcal{F})} dis(\bar{s},\bar{s}'), \tag{20}$$

where $kNN(\bar{s},\mathcal{D}\cup\mathcal{F})$ denotes the k-nearest neighbors of $\bar{s}$ in the feasible and failed demonstration sets, and $dis(\bar{s},\bar{s}')$ is the distance metric. In the experiments, we set $k=3$ and use Euclidean distance. The trajectory-level diversity score is

$$D(\tau) = \frac{1}{|\tau|} \sum_{(s,a)\in\tau} d(\bar{s}). \tag{21}$$

The final query score is

$$Q(\tau) = U(\tau) + w_d D(\tau), \tag{22}$$

where w_d is the diversity weight. In each active learning iteration, the top N_q trajectories with the highest scores are queried and then added to the training set with expert-provided trajectory-level labels.

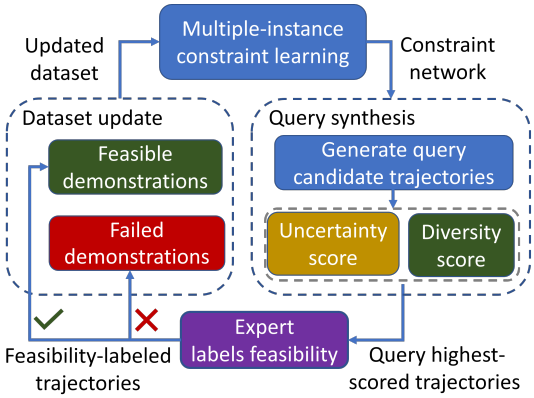


Fig. 3. The framework of the proposed active constraint learning method.The constraint network is first trained with the initial demonstration sets. Next, in each following iteration, the algorithm generates N_s query candidate trajectories from random starting points near the demonstrations, and query top N_q trajectories with the highest scores $Q(\tau)$. The queried trajectory, after being labeled by the expert, will be appended to either the feasible demonstration set $\mathcal{D}$ or the failed demonstration set $\mathcal{F}$ accordingly. The constraint model will be refined with the updated dataset. The algorithm stops when reaching some maximum iterations or when the user is satisfied with recent queried trajectories.

C. Query Candidate Trajectory Generation

Previous section discusses about picking the most informative trajectories from a given candidate trajectory set $\mathcal{T}$. This section focuses on the generation of such a trajectory set. Generally speaking, these trajectories are expected to perform the same task as in both demonstrations. Depending on whether a reward function is known, one can either use an imitation learning (IL) approach or an RL approach to generate these trajectories.

The IL approaches directly fit a policy model from demonstrations and does not require a known reward function. However, while the policy fitted from the demonstrations may be feasible, they do not explicitly model the underlying constraints. Therefore, the learned policy may not consistently satisfy them during deployment [5], [11]. Moreover, IL policies are often task-specific and cannot be directly adapted to new tasks with different goals but the same constraints, whereas an explicitly learned constraint model can generalize across such tasks [5], [11].

While our MICL method and active learning framework are compatible with most IL methods, this work specifically considers a Dynamical System (DS) policy [39], which is widely used in robotics application. DS policy is a Gaussian Mixture Regression model fitted from feasible demonstrations with additional stability conditions. The learned DS can be further modulated by multiplying a rotation matrix to exhibit constraint-compliant behaviors.

Dynamical System Policy: DS initially fits an unconstrained nominal policy and later modulates nominal policy

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

with the constraint network. The nominal dynamical system policy is represented as (23), where s_g is the goal state and matrix $A(s)$ is a Gaussian Mixture Regression model learned from feasible demonstrations and preserves stability, following the Linear Parameter Varying Dynamical System (LPVDS) method [40]. Alternatively, for task of reaching a goal in the shortest path, a linear DS $A(s) = I$ is directly applicable without fitting.

$$\pi_n(s) \equiv \dot{s} = A(s)(s - s_g) \tag{23}$$

In each iteration, we generate the candidate trajectory set $\mathcal{T}$ using a modulated DS, rather than the nominal DS. Given a constraint network $\zeta_\theta(s)$, the modulated DS is a constraint-compliant policy obtained by applying a rotation matrix $M(s)$ on the nominal DS [41]:

$$\begin{aligned}
&\pi(s) = M(s)\pi_n(s), \quad \text{where} \\
&M(s) = E(s)D(s)E^{-1}(s), \\
&E(s) = [r(s), e_1(s), e_2(s), ...e_{n-1}(s)], \\
&n(s) = \frac{d\zeta(s)}{ds}, n(s) \perp e_1(s) \perp e_2(s) \perp ...e_{n-1}(s) \\
&D(s) = diag[\lambda_1(s), \lambda_2(s), \lambda_3(s)..., \lambda_n(s)], \\
&\lambda_1(s) = 1 - \frac{1}{\Gamma(s)}, \lambda_{2,3,...,n}(s) = 1 + \frac{1}{\Gamma(s)}, \\
&\Gamma(s) = 1 + 10 * (\zeta(s) - 0.5), \text{for } \zeta(s) \geq 0.5
\end{aligned} \tag{24}$$

where $n(s)$ is the normal vector to the constraint boundary, $e_i(s)$ are the corresponding tangent vectors, $r(s)$ is the vector towards a reference point inside the obstacle, $\lambda_i(s)$ modifies the velocity along different directions, and $\Gamma(s)$ specifies the relative distance of current state s to the constraint boundary. The modulated DS is defined only on learned feasible region where $\zeta(s) \geq 0.5$. For any feasible state s close to the learned constraint boundary, $\Gamma(s)$ approaches 1 and $\lambda_1(s)$ approaches 0. Since $\lambda_1(s)$ scales the velocity component along the first basis direction $r(s)$, the modulation suppresses motion along $r(s)$, which points toward the infeasible region, while preserving motion along the remaining basis directions.

The modulated DS guarantees that all generated trajectories satisfy the learned constraints [39]. Additionally, if the true constraint is a star-shaped obstacle and the nominal policy is linear, the modulated policy also remains globally stable [41]. These benefits make the modulated DS more reliable if deployed in the real world. In each iteration, we generate the query candidate set from the modulated DS rather than the nominal DS.

Reinforcement Learning Policy: If the reward function is available, one can train an RL policy to generate query candidate trajectories. The RL policy is a fully-connected network trained by optimizing the reward function while satisfying the learned constraint. To achieve this, we adopt a straightforward penalty method [42] for constrained policy optimization. As shown in (25), we reshape the original reward by incorporating a penalty term proportional to the constraint violation into the reward function, discouraging the policy from visiting infeasible states. Here, w_p is a penalty weight. The RL policy is learned with the reshaped reward using the standard PPO algorithm [43].

$$\begin{aligned}
&r'(s,a) = r(s,a) - w_p c_\theta(s,a), \quad \text{where} \\
&c_\theta(s,a) = \begin{cases} 0 & \text{if } \zeta_\theta(\bar{s}) > 0.5 \text{ (feasible)} \\ 1 & \text{if } \zeta_\theta(\bar{s}) \leq 0.5 \text{ (infeasible)} \end{cases}
\end{aligned} \tag{25}$$

Readers familiar with ICRL methods may notice that active constraint learning with an RL policy resembles the structure of many ICRL approaches [5]. However, a key difference is that our method does not assume the demonstrations are optimal. Consequently, we cannot rely on the reward signal to infer constraint satisfaction and must instead infer feasibility purely from the expert-provided labels.

VI. Experiment

The experimental section consists of two parts. The first part studies the performance of the proposed MICL for learning from fixed demonstration sets without active learning (Section IV), and second part subsequently verifies the active learning framework with expert's feedback (Section V). All code and datasets are available at https://github.com/epfl-lasa/Actively_Learning_Constraints_from_Failed_Demonstrations.

A. Constraint Learning from Fixed Demonstration sets

Baselines: Five constraint learning methods are compared here: (1) **MICL (ours)**; (2) **BC (Binary Classification [11])**, a straightforward method that simply takes all states from feasible demonstrations as feasible data, and all states from failed demonstrations as infeasible to train a binary classification network with standard cross entropy loss. (3) **MECL2 (Maximum Entropy Constraint Learning [5] with adaptation)**. The original MECL is popular ICRL method capable of learning constraint networks, but it is specifically designed to learn from solely feasible demonstrations. To enable learning from failed data, we modify its gradient update of the constraint network as follows:

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta) = &\mathbb{E}_{\tau\sim\mathcal{D}}\left[\mathbb{E}_{(s,a)\sim\tau}\left[\nabla_\theta \log \zeta_\theta(\bar{s})\right]\right] \\
&- \mathbb{E}_{\tau\sim\mathcal{F}}\left[\mathbb{E}_{(s,a)\sim\tau}\left[\nabla_\theta \log \zeta_\theta(\bar{s})\right]\right],
\end{aligned} \tag{26}$$

This adapted gradient is based on Equation (7) of [5]. (4) **CT (Constraint Transformer [44])**. CT is a recent transformer-based constraint learning method. The original CT framework assumes access to a known reward function, maintains a world model, and performs iterative policy-constraint updates. Since these components are unavailable in our fixed-dataset setting, we extract only the constraint transformer module from the authors' open-source implementation, and train constraint non-iteratively on our fixed feasible/failed datasets. **(5) DCT (Decoupled Constraint Transformer)**. DCT is a variant of CT that keeps the transformer module to assign trajectory-level importance weights to states, but uses a separate point-wise MLP head as the constraint function that outputs state-level feasibility prediction.

A list of important hyperparameters of our method in all tasks is given in Table I in Appendix B. For the neural network architectures, the number of hidden units in each layer is mentioned. The hidden activation functions in all networks are Leaky ReLU. The output activation function of the constraint

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

network is the sigmoid function, whereas that of the policy and value network is tanh. All baselines adopt identical network architecture except for CT and DCT.

Metrics: During evaluation we introduce two metrics: 1) the IoU (the intersection over union) index, which measures the correctness of the learned constraints. We uniformly sample points in the state space, and compute IoU as (Number of points which are both predicted to be infeasible and truly infeasible) divided by (Number of points either predicted to be infeasible or truly infeasible); a higher IoU indicates a more accurate constraint. 2) the constraint violation rate, the per-step true constraint violation rate of the policy learned (or modulated) from the learned constraint. In general, a lower violation rate indicates a more accurate constraint. It is noteworthy that the two metrics assess constraint learning from different perspectives, and a low IoU does not necessarily imply a high violation rate. For instance, a constraint network that naively classifies most states—including many feasible ones—as infeasible may have a low IoU but still result in a very low violation rate.

Tasks: Five constraint learning tasks are studied.

1) 2D Reaching: a toy goal-reaching task in 2D position space, where a point robot is initialized randomly and tasked to reach a goal state while staying away from an unknown irregular obstacle composed of two ellipses as shown in Fig. 4. The control variables are 2D velocities. The demonstration sets, visualized in Fig. 4, consist of 11 feasible demonstrations and 3 failed demonstrations, both generated using DS policy.

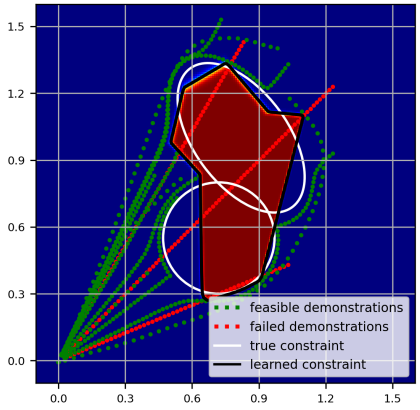


Fig. 4. True constraint and an example of learned constraint with the proposed method of the 2D reaching task. The two white ellipses represent the boundary of the true constraint. The heat map is a visualization of the learned constraint ζ_θ, where the red region of $\zeta = 0$ represents infeasibility, while the blue regions of $\zeta = 1$ indicate feasibility. The feasible and failed demonstrations are shown in green and red dotted lines, respectively.

2) GlassAvoid: This task considers a fragile glass avoidance scenario in Fig. 5, where a robot arm moves a heavy object to a fixed target near four fragile glass cups. For safety, the end-effector's center of mass must remain outside four cylindrical regions surrounding the glasses. The state consists of the 3D end-effector position, the controls are the corresponding velocities. Failed trajectories may pass over the glasses, creating a safety risk.

Although the true unsafe regions are composed of cylinders, this geometry is not given to the agent and must be inferred from demonstrations. This makes the setting more challenging than studies such as [13], which assume known obstacle geometry and infer only a few geometric parameters.

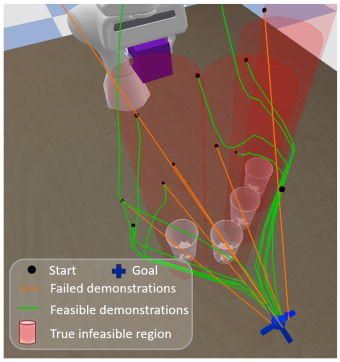


Fig. 5. Visualization of the GlassAvoid task. The robot end-effector is required to reach the target while avoiding the unknown infeasible regions around the glass cups. A demonstration is feasible if it reaches the goal without entering any infeasible region, and failed if it enters at least one infeasible region. Green and red trajectories show feasible and failed demonstrations, respectively. The translucent red cylinders show the ground-truth infeasible regions for visualization only.

The dataset contains 30 feasible and 20 failed demonstrations. Feasible demonstrations are generated by a DS policy modulated with the true constraints, while failed demonstrations are generated by a linear unmodulated DS policy. The failed demonstrations cover only a limited part of the state space, constraint recovery is not expected to be perfect, leaving room for the active constraint learning approach introduced in the next section.

3) SphereSpeed: Unlike the previous tasks with purely geometric position constraints, SphereSpeed requires learning position, speed magnitude, and speed-direction constraints simultaneously. The end-effector moves inside a spherical workspace whose center is the origin. The constraint network takes the augmented feature vector $\bar{s} = (x, y, z, r, v, \alpha)$ as input, where $p = (x, y, z)$ is the current position, $r = \|p\|$, v is the speed magnitude, and α is the cosine alignment between the speed direction and the inward radial direction.

The position constraint requires the current state to remain inside a sphere, $r \leq 0.13$. The speed constraint is position dependent: $v \leq 0.65 + 0.45 \operatorname{clip}\left(\frac{|x|}{0.13}, 0, 1\right)$. A radius-dependent speed direction constraint is also present. For $r \leq 0.09$, the velocity direction is unrestricted. For $0.09 < r \leq 0.13$, outward motion is increasingly restricted, with the maximum allowed angle from the inward radial direction linearly decreasing to 150° at the boundary.

To collect feasible demonstrations that explore within the sphere, we use a temporally smoothed random-walk policy equipped with a projection-based safety filter [2] to generate 100 feasible demonstrations. We then remove this safety filter and use the same random-walk policy to generate 100 failed demonstrations.

[2]The safety filter projects each sampled velocity command back into the feasible set via constrained optimization when it would otherwise violate the constraint.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

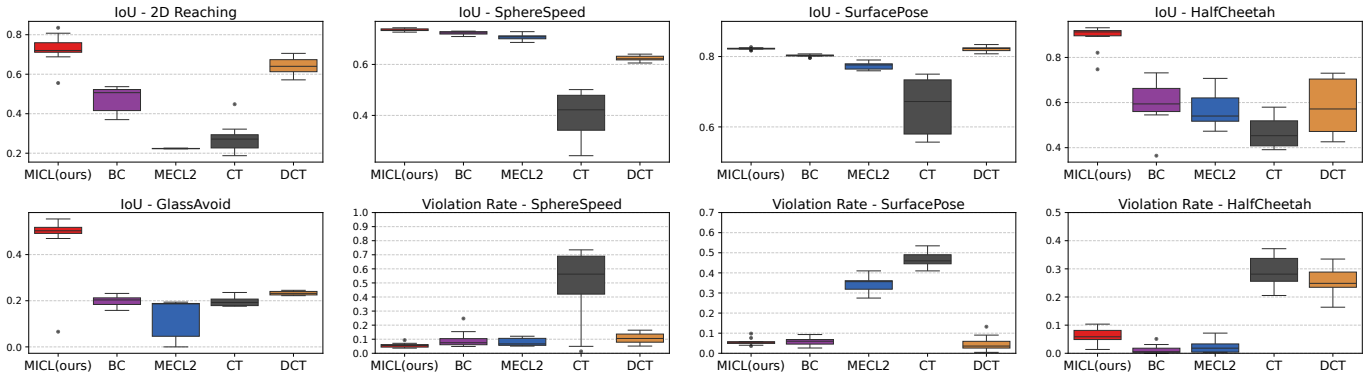


Fig. 6. Comparison of constraint learning from fixed demonstrations across five tasks. The first column reports the IoU index for 2D Reaching and GlassAvoid. The remaining columns report the IoU index (top) and the true constraint violation rate of the learned policy (bottom) for SphereSpeed, SurfacePose, and HalfCheetah. IoU is higher better, while violation rate is lower better. All results are based on 10 independent runs with the same demonstrations but different network initializations.

4) SurfacePose: SurfacePose (shown in Fig. 7) evaluates constraint learning in a 6D pose-control task with both position and orientation constraints. The agent controls an end-effector pose $s = (x, y, z, \phi, \theta, \psi)$, where (x, y, z) is the Cartesian position and (ϕ, θ, ψ) are roll-pitch-yaw angles. The action is a 6D pose increment, and the constraint network takes the full pose as input. The position constraint requires the end-effector to stay close to an unknown sinusoidal surface:

$$z_s(x,y) = 0.11\left[0.55\sin\left(1.2\frac{x-0.40}{0.11}\right) + 0.35\cos\left(\frac{y+0.22}{0.11}\right)\right]. \tag{27}$$

A pose violates the position constraint if $|z - z_s(x,y)| > 0.015$. The orientation constraint requires the end-effector to remain nearly upright: it must stay within 8° of the world-up direction.

150 feasible demonstrations and 150 failed demonstrations are generated using similar random walker policy as in SphereSpeed.

5) Blocked Halfcheetah: This is a higher-dimensional environment widely used in prior studies [5], [11]. As shown in Fig. 7, the agent controls a two-legged cheetah robot by applying torques to its joint motors, and is rewarded based on the forward distance traveled between consecutive time steps. Unlike prior works [5], [13], our experiments in section VI-A does not assume knowing this reward function. The true constraint is the red wall that blocks the region with x-coordinate ≤ -1, so the robot is only allowed to move in the region with x-coordinate between -1 and ∞. The input to the constraint network is an 18-dimensional state space, including positional and angular values of different body parts of the cheetah, followed by their velocities, making constraint extraction challenging due to this high-dimensional space.

To collect demonstration sets, we trained a highly suboptimal RL policy using the true constraint and obtained 20 feasible and 20 failed trajectories by sampling from this policy.

Results: The IoU scores of all tasks are shown in Fig. 6. For the higher-dimensional tasks, namely SphereSpeed, SurfacePose, and HalfCheetah, we also report the violation rates of policies trained using the learned constraints to further evaluate constraint quality.

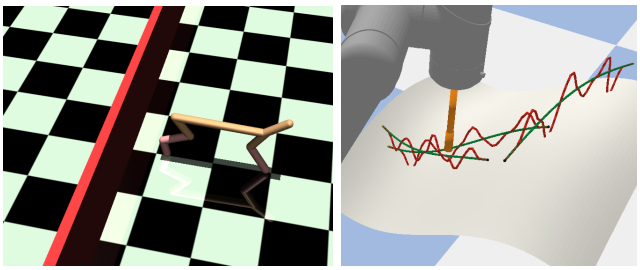


Fig. 7. Visualization of the Blocked HalfCheetah and SurfacePose tasks. Left: in Blocked HalfCheetah, the agent is encouraged to run fast while avoiding the red wall. Right: in SurfacePose, the robot end-effector must move along an unknown curved surface while maintaining an upright orientation; Trajectories are examples of demonstrations, feasible in green and failed in red.

Overall, MICL achieves the best IoU on all tasks except SurfacePose, where MICL and DCT perform similarly. On SurfacePose, the median IoU of MICL and DCT is 0.8225 and 0.8228, respectively. For violation rate, MICL obtains a median of 0.0549, while DCT obtains a median of 0.0359. These results indicate that MICL consistently learns accurate aggregate feasibility functions across diverse environments, including the multi-constraint SphereSpeed and SurfacePose tasks. For the higher-dimensional tasks, MICL also achieves the lowest or second-lowest violation rate, showing that the learned constraints remain useful for downstream policy execution.

Among the baselines, BC performs reasonably well despite its simplicity, but its performance is less consistent across tasks because it treats all states in failed demonstrations as infeasible. CT generally performs poorly, suggesting that the original trajectory-level transformer cost model may be unsuitable for learning state-level constraint. DCT improves over CT by decoupling the trajectory-level transformer module from the point-wise constraint output, and achieves its strongest performance on SurfacePose. We hypothesize that this is because SurfacePose has substantially more demonstrations

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

than the other tasks, making it more favorable for transformer-based architectures.

The violation-rate results also highlight the difference between constraint boundary accuracy and policy-level safety. In HalfCheetah, MECL2 and BC obtain low violation rates but relatively poor IoU, mainly because they learn overly conservative constraints and classify many truly feasible states as infeasible.

Sensitivity to hyperparameters and number of demonstrations: We further study the sensitivity of MICL performance to the number and ratio of two demonstration sets and the LSE pooling hyperparameter ω.

In Fig. 8, we report the learning results when varying the number of feasible (or failed) demonstrations while keeping the number of failed (or feasible) demonstrations fixed. Results indicate that increasing either the number of feasible or failed demonstrations generally improves IoU while also reducing the variance across runs. However, adding more failed demonstrations beyond a certain number yields relatively minor improvements.

MICL introduces a smoothness hyperparameter ω in the LSE pooling layer, controlling the smoothness of the LSE aggregation and consequently the tightness of the learned constraint. Since ω may affect learning performance, we analyze its sensitivity in Fig. 9 on the 2D Reaching and HalfCheetah tasks. The results show that both tasks exhibit a stable high-performance range rather than requiring an isolated single value.

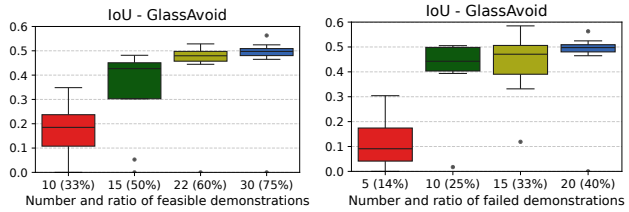


Fig. 8. Effect of the number and ratio of two types of demonstrations on constraint learning in the GlassAvoid task. Left: IoU index of the learned constraint when the number of failed demonstrations is fixed to 20 and the number of feasible demonstrations varies. Right: IoU index when the number of feasible demonstrations is fixed to 30 and the number of failed demonstrations varies. Each boxplot is based on 8 independently sampled demonstration subsets.

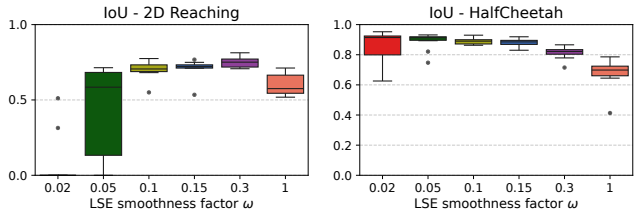


Fig. 9. Sensitivity analysis with respect to the LSE smoothness factor in the 2D Reaching and HalfCheetah tasks. Each boxplot is based on 10 independent runs.

B. Active Constraint Learning from Expert Feedback

This subsection investigates the effectiveness of the proposed active constraint learning framework, in which new trajectories are iteratively generated and queried to refine the constraint model.

Experiment setup: We compare three active query strategies with a random query baseline: Monte Carlo Dropout, temperature-scaled entropy, and Bayesian Active Learning by Disagreement, abbreviated as Dropout, TS-entropy, BALD, and Random, respectively. Dropout and TS-entropy use the combined uncertainty-diversity query score in (22), where the uncertainty component is computed differently for each method and the diversity component is consistently computed using the k-nearest-neighbor distance in (20). BALD uses the trajectory-level mutual-information score with zero diversity weight, while Random selects query trajectories uniformly. After the demonstration sets are updated, all methods use MICL with LSE pooling to update the constraint network.

Since Dropout and BALD rely on dropout layers for query-time stochastic predictions, we also apply the same dropout layers for Random and TS-entropy to ensure a fair comparison, but the dropout is disabled during inference and is not used for query scoring.

Compared to Section VI-A, we partially reduce the number or quality of initial demonstrations to leave more room for active learning. In this section, 2D Reaching uses 11 feasible and 3 failed demonstrations, and GlassAvoid uses 30 feasible and 11 failed demonstrations. For both tasks, the failed demonstrations are intentionally selected to be spatially concentrated, in order to simulate realistic scenarios where constraint violations are initially observed in only a limited region. Halfcheetah, in particular, starts with only 1 feasible and 1 failed demonstration, making it a strongly data-scarce scenario. To be fair, all methods are trained using the same initial demonstration sets. A list of important hyperparameters is summarized in Table II of Appendix B.

Results: The learning results across three tasks are shown in Fig. 10. For the 2D Reaching and GlassAvoid tasks, a DS policy is adopted. While for Halfcheetah task, we require the true reward function is known and an RL policy is performed. In the 2D Reaching and GlassAvoid tasks, all methods show stable improvements in constraint learning accuracy as the number of queries increases. Notably, the three active querying strategies all outperform the random querying baseline in both IoU and violation rate, demonstrating the effectiveness of active learning. The active query strategies also exhibit smaller variance than Random. For example, in GlassAvoid, the relative standard deviations of the IoU scores are 2.95%, 3.75%, and 4.53% for Dropout, TS-entropy, and BALD, respectively, while the relative standard deviation of Random is much higher (15.68%). Among three active learning method, BALD achieves overall best mean IoU metrics.

To provide an intuitive understanding of the active constraint learning process, we present the visualization of a typical learning process with the TS-entropy method in Fig. 11. The figure illustrates how both the constraint function and the model's uncertainty evolve over iterations. At each iteration,

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

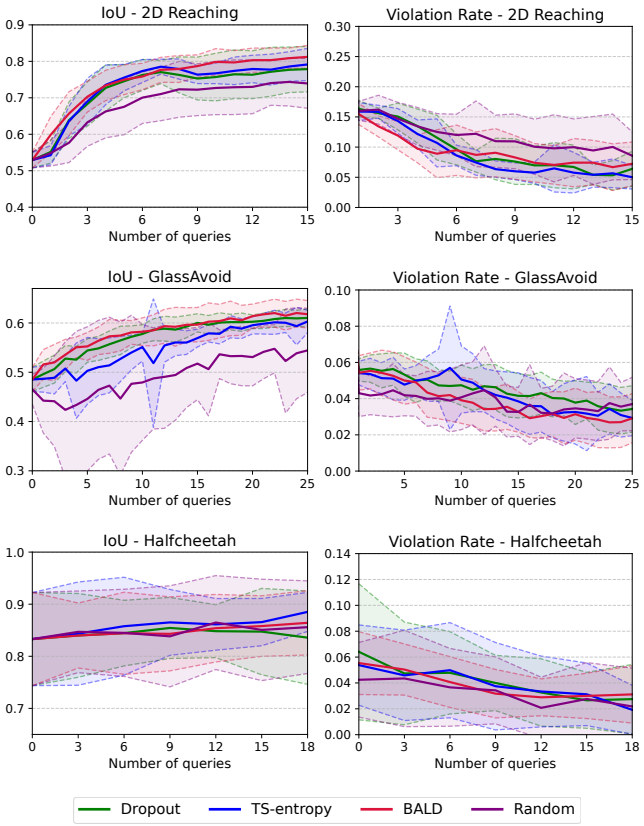


Fig. 10. Comparison of four active constraint learning methods across three tasks: 2D Reaching (top), GlassAvoid (middle), and Halfcheetah (bottom). In each row, the left figure shows the IoU index (higher is better) of the learned constraint, and the right shows the true constraint violation rate of the policy trained with it (lower is better). The x-axis indicates the number of queries. All results are based on the mean of 14 independent runs; shaded areas represent one standard deviation.

the agent generates a set of candidate trajectories and selects the most informative ones based on a combined uncertainty-diversity score to query the expert. The constraint model is then refined using the newly labeled data.

In the Halfcheetah task, TS-entropy again achieves the highest IoU, while BALD and Random exhibit similar performance. Moreover, we observe that the advantage of active querying over random querying is less pronounced in this task. One possible explanation is that, in the Halfcheetah environment, the true constraint is simply a wall, in contrast to the complex obstacle geometries in the previous tasks. As a result, the candidate trajectories generated from different initial states tend to be similar—straightly moving rightward, making the specific choice of which trajectory to query less impactful. This task also has larger variance. This is mainly because the initial dataset contains only one feasible and one failed trajectory, so each new query can substantially change the learned constraint. In addition, HalfCheetah uses an RL policy during active learning, whose stochastic training introduces extra variance in candidate generation and policy performance.

Overall, these results show that the proposed active constraint learning framework can progressively improve the quality of the learned constraint through trajectory-level expert feedback. In general, active query strategies outperform random querying in both IoU and policy violation rate. These results also indicate that the proposed framework is not tied to a specific query strategy and can incorporate different principled active-learning criteria.

Effect of diversity query and sensitivity to diversity weight. We further investigate the effect of adding the diversity term to the query score. Fig. 12 shows the performance of the Dropout method with different diversity weights. The results suggest that introducing a moderate diversity weight ($w_d = 5$) improves performance compared with using uncertainty alone ($w_d = 0$), especially in the later stage of training. This improvement is attributed to the ability of diversity-aware querying to avoid repeatedly selecting similar trajectories and to better explore underrepresented regions of the state space. On the other hand, excessively large weights ($w_d = 20$), which resemble pure diversity-based querying, can slightly degrade performance, likely due to oversampling from outlier regions. These findings confirm the benefit of combining diversity and uncertainty in query selection.

Effect of mislabeled query data We also investigate the robustness of active constraint learning to label noise, which is inevitable in real-world applications where human experts may mislabel the feasibility of trajectories during query process. To simulate this, we randomly flip the labels of queried trajectories during active learning with a fixed probability, referred to as the mislabeling rate, denoted by r_w. Note that the flipping is applied only to queried trajectories, while the initial demonstration sets remain unaltered. Fig. 13 shows the learning performance with Dropout method under different mislabeling rate r_w. The results suggest that our method retains reasonable performance under varying levels of label noise, demonstrating a degree of robustness. However, when the noise level becomes relatively high (e.g., 10% or 20%), the performance plateaus even with an increased number of queries, indicating that further data collection cannot compensate for the degradation caused by noisy supervision.

VII. Conclusions and Limitations

This paper proposes a novel constraint learning method MICL that actively learns both feasible and failed demonstrations, the latter of which existing methods are generally unable to exploit and therefore discard. MICL employs a constraint network to predict the state-level feasibility, with a specialized pooling layer that aggregates these predictions to produce trajectory-level feasibility estimates, which are optimized to match the ground-truth labels of the demonstrations (feasible or failed). Theoretical analysis proves that MICL can identify infeasible states with bounded errors, and thus recover the underlying constraint function.

Furthermore, an active learning framework is also incorporated that iteratively refines the constraint model by querying the expert on the feasibility of newly generated trajectories, where the queries are selected based on a combined uncertainty-diversity score.

We validate the proposed method in five simulated constraint learning tasks with different dimensions and constraint structures. The results show that MICL outperforms

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

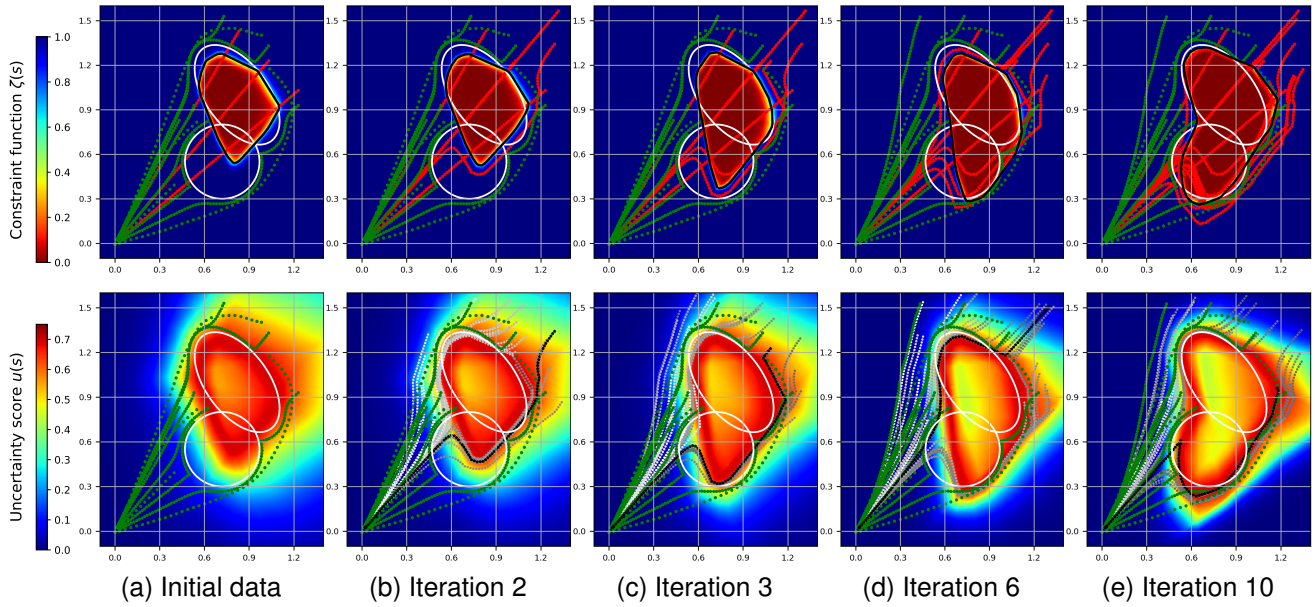


Fig. 11. Visualization of five representative iterations in the active constraint learning process, where each column corresponds to a certain iteration. **Top row**: Learned constraint function $\zeta(s)$, where red regions indicate infeasible states and blue regions indicate feasible ones. White ellipses denote the ground-truth constraint boundaries. Green dots are feasible demonstrations and feasible queried trajectories; red dots are failed demonstrations and queries. **Bottom row**: Uncertainty score $u(s)$ estimated by TS-entropy, with red indicating high uncertainty. White-to-black dots represent query candidate trajectories, where darker color corresponds to higher uncertainty-diversity scores. Among them, the black dots have the highest score and is the one selected for querying in the current iteration. Initially, only the upper ellipse constraint is captured due to limited failed demonstrations. As active learning progresses, the lower ellipse constraint is gradually discovered.

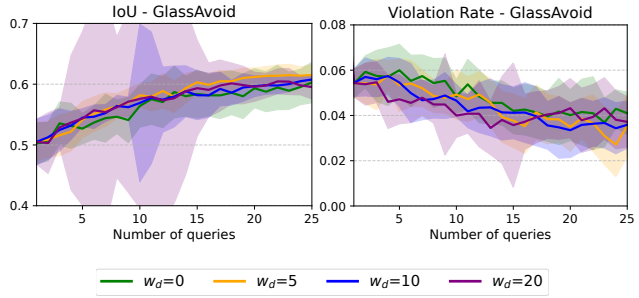


Fig. 12. Comparison of active constraint learning with different diversity weights w_d in the GlassAvoid task. Left: The IoU index (higher is better) of the learned constraint. Right: True constraint violation rate of the policy trained with the learned constraint (lower is better). All the curves are based on 8 independent runs using dropout-based uncertainty estimation. The shaded area represents one standard deviation.

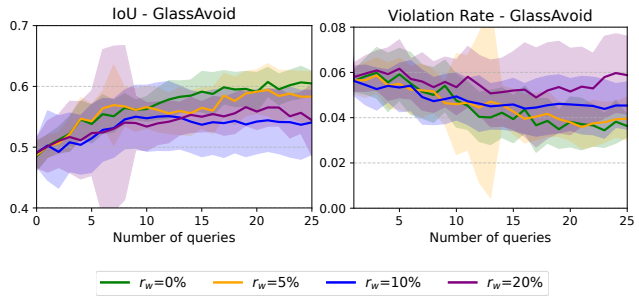


Fig. 13. Impact of mislabeling rate r_w on active constraint learning in the GlassAvoid task. Left: The IoU index (higher is better) of the learned constraint. Right: True constraint violation rate of the policy trained with the learned constraint (lower is better). All results are means over 14 independent runs. The shaded area represents one standard deviation.

or matches the strongest baselines in fixed-demonstration constraint learning. The active learning framework also progressively improves constraint accuracy through additional trajectory-level expert feedback.

A main limitation of this work is that the proposed framework is evaluated only in simulated tasks. Extending to more realistic and complex robotic settings, especially physical systems with real human feedback, is an important direction for future work. Another limitation is that query candidates are generated using policies guided by the current constraint network. When this network is highly inaccurate, the candidate pool may be unreliable. Future work will study more robust candidate-generation strategies.

APPENDIX A
PROOF OF THEOREM 1

Proof: Fix an infeasible trajectory τ. If all $\bar{s} \in K_\tau$ are predicted feasible, then the trajectory is predicted feasible; hence

$$\begin{aligned}
\Pr\left[\hat{y}(\tau)=1 \mid \tau\right] &= \Pr\Big(\bigcap_{\bar{s}\in K_\tau}\{\hat{y}(\bar{s})=1\} \;\Big|\; \tau\Big) && (28)\\
&\geq \underline{\alpha}\prod_{\bar{s}\in K_\tau} q_{\bar{s},\tau} && \text{(Assumption 2)}\\
&\geq \underline{\alpha}\,(\rho\,\bar{q}_\tau)^{|K_\tau|} && \text{(Assumption 1)}\\
&\geq \underline{\alpha}\,(\rho\,\bar{q}_\tau)^{M} && \text{(since } |K(\tau)|\leq M\text{)}
\end{aligned}$$

Expanding the trajectory-level FF rate by taking expectation over infeasible trajectories,

$$\mathcal{E}^{\mathrm{FF}}_{\mathrm{traj}} = \mathbb{E}_\tau\left[\Pr(\hat{y}(\tau)=1 \mid \tau)\right] \geq \underline{\alpha}\,\mathbb{E}_\tau\left[(\rho\,\bar{q}_\tau)^{M}\right]. \qquad (29)$$

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

Finally, since $x \mapsto x^M$ is convex on $[0,1]$, Jensen's inequality yields

$$\mathbb{E}_\tau[\bar{q}_\tau^M] \geq \Big(\mathbb{E}_\tau[\bar{q}_\tau]\Big)^M = \Big(\mathbb{E}_{\bar{s},\tau}[q_{\bar{s},\tau}]\Big)^M = \big(\mathcal{E}^{\text{FF}}_{\text{state}}\big)^M. \quad (30)$$

Taking (30) into (29),

$$\mathcal{E}^{\text{FF}}_{\text{traj}} \geq \underline{\alpha}\big(\rho\mathcal{E}^{\text{FF}}_{\text{state}}\big)^M. \quad (31)$$

Rearranging gives (16). ■

APPENDIX B
HYPERPARAMETERS

TABLE I
MICL HYPERPARAMETERS.

Environment	Layers	learning rate	epochs	LSE ω
2D Reaching	32,32	1×10^{-3}	4800	0.2
GlassAvoid	32,32	1×10^{-3}	8000	0.05
HalfCheetah	4,4	1×10^{-3}	4800	0.05
SurfacePose	64,64	3×10^{-3}	3000	0.5
SphereSpeed	128,128	1×10^{-3}	4500	0.26

APPENDIX C
COMPUTATIONAL COMPLEXITY ANALYSIS

We analyze the computational complexity of the proposed method by separating the analysis into two stages: (1) constraint learning from a fixed demonstration dataset, and (2) the active constraint learning.

Let N denote the number of trajectories in the initial demonstration dataset (including both feasible and failed ones), and T denote the average length of each trajectory. Each datapoint (i.e., state-action pair) has a dimension of d. Let f_{NN} denote the computational cost of a single forward and backward pass through the network per sample, which depends on the specific network architecture and the dimension of input. Then, training the constraint network for E epochs on the fixed dataset incurs a total cost of: $O(ENTf_{\text{NN}})$.

In the active learning setting, the constraint network is updated iteratively as newly labeled trajectories are added. Each iteration involves the following components:

- *Policy Update:* For the DS policy, modulation does not require explicit optimization. For the RL policy, the cost depends on the specific RL algorithm used. We denote the policy update cost uniformly as f_l.
- *Query Candidate Generation:* N_s candidate trajectories of length T are generated using the current policy. Let f_π be the cost of a single policy inference and f_e the cost of one environment simulation step. The total generation cost is: $O(N_sT(f_\pi + f_e))$.
- *Query Scoring and Selection:* Taking the TS-entropy method as an example, computing uncertainty scores for all candidates costs $O(N_sTf_{\text{NN}})$. Computing diversity scores via k-nearest neighbors has a cost of $O(N_sNT^2d)$. The cost of selecting top-scored trajectories is linear in N_s and thus negligible.
- *Constraint Network Retraining:* Retraining the constraint network on N trajectories (each of length T) for E epochs costs: $O(ENTf_{\text{NN}})$.

Assuming L active learning iterations, the total complexity is:

$$O\Big(L\big[f_l + N_sT(f_\pi + f_e + f_{\text{NN}} + NTd) + ENTf_{\text{NN}}\big]\Big) \quad (32)$$

Although the overall complexity is higher than that of conventional active learning methods operating on static datasets, the structure of the cost remains comparable in nature.

REFERENCES

[1] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[2] B. Peng, Y. Mu, J. Duan, Y. Guan, S. E. Li, and J. Chen, "Separated proportional-integral lagrangian for chance constrained reinforcement learning," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 193–199.

[3] B. Peng, J. Duan, J. Chen, S. E. Li, G. Xie, C. Zhang, Y. Guan, Y. Mu, and E. Sun, "Model-based chance-constrained reinforcement learning via separated proportional-integral lagrangian," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 466–478, 2022.

[4] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations," in *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer, 2020, pp. 228–245.

[5] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7390–7399.

[6] B. Peng and A. Billard, "Positive-unlabeled constraint learning for inferring nonlinear continuous constraints functions from expert demonstrations," *IEEE Robotics and Automation Letters*, 2024.

[7] C. Pérez-D'Arpino and J. A. Shah, "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4058–4065.

TABLE II
MICL ACTIVE CONSTRAINT LEARNING HYPERPARAMETERS.

	2D Reaching	GlassAvoid	Halfcheetah
Constraint network layers	32, 32	32, 32	4, 4
Constraint learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}
Training epoch E per iteration	500	1000	1200
Dropout Rate	0.05	0.05	0.05
LSE smoothness factor ω	0.2	0.1	0.05
Number of initial feasible demonstrations	11	30	1
Number of initial failed demonstrations	3	11	1
Number of query candidates N_s per iteration	30	30	30
Number of queries N_q per iteration	1	1	3
Diversity score weight w_d	20	5	0
Number of forward passes (Dropout/BALD only)	50	50	50
Temperature T (TS-Entropy only)	15	15	15

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2026.3731354

[8] G. Subramani, M. Zinn, and M. Gleicher, "Inferring geometric constraints in human demonstrations," in *Conference on Robot Learning*. PMLR, 2018, pp. 223–236.

[9] G. Chou, H. Wang, and D. Berenson, "Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3827–3834, 2022.

[10] K. C. Stocking, D. L. McPherson, R. P. Matthew, and C. J. Tomlin, "Maximum likelihood constraint inference on continuous state spaces," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8598–8604.

[11] G. Liu, Y. Luo, A. Gaurav, K. Rezaee, and P. Poupart, "Benchmarking constraint inference in inverse reinforcement learning," *International Conference on Learning Representations*, 2023.

[12] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3682–3690, 2020.

[13] ——, "Learning parametric constraints in high dimensions from demonstrations," in *Conference on Robot Learning*. PMLR, 2020, pp. 1211–1230.

[14] D. H. Grollman and A. G. Billard, "Robot learning from failed demonstrations," *International Journal of Social Robotics*, vol. 4, pp. 331–342, 2012.

[15] D. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *International Conference on Learning Representations*, 2020.

[16] A. Glazier, A. Loreggia, N. Mattei, T. Rahgooy, F. Rossi, and B. Venable, "Learning behavioral soft constraints from demonstrations," *arXiv preprint arXiv:2202.10407*, 2022.

[17] D. L. McPherson, K. C. Stocking, and S. S. Sastry, "Maximum likelihood constraint inference from stochastic demonstrations," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 1208–1213.

[18] S. Xu and G. Liu, "Uncertainty-aware constraint inference in inverse constrained reinforcement learning," in *The Twelfth International Conference on Learning Representations*, 2023.

[19] G. Quan, G. Liu *et al.*, "Learning constraints from offline demonstrations via superior distribution correction estimation," in *Forty-first International Conference on Machine Learning*, 2024.

[20] S. G. Subramanian, G. Liu, M. Elmahgiubi, K. Rezaee, and P. Poupart, "Confidence aware inverse constrained reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2024, pp. 14 491–14 512.

[21] A. Gaurav, K. Rezaee, G. Liu, and P. Poupart, "Learning soft constraints from constrained expert demonstrations," in *International Conference on Learning Representations*, 2023.

[22] D. Papadimitriou, U. Anwar, and D. S. Brown, "Bayesian methods for constraint inference in reinforcement learning," *Transactions on Machine Learning Research*, 2022.

[23] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems*, 2017.

[24] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[25] E. Biyik and D. Sadigh, "Batch active preference-based learning of reward functions," in *Proceedings of the 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 2018, pp. 519–528.

[26] X. Liang, K. Shu, K. Lee, and P. Abbeel, "Reward uncertainty for exploration in preference-based reinforcement learning," in *International Conference on Learning Representations*, 2022.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[28] M. Waqas, S. U. Ahmed, M. A. Tahir, J. Wu, and R. Qureshi, "Exploring multiple instance learning (mil): A brief survey," *Expert Systems with Applications*, p. 123893, 2024.

[29] X. Wang, Y. Yan, P. Tang, X. Bai, and W. Liu, "Revisiting multiple instance neural networks," *Pattern recognition*, vol. 74, pp. 15–24, 2018.

[30] S. Sabato and N. Tishby, "Multi-instance learning with any hypothesis class," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2999–3039, 2012.

[31] G. Doran and S. Ray, "Multiple-instance learning from distributions," *Journal of Machine Learning Research*, vol. 17, no. 128, pp. 1–50, 2016.

[32] A. Blum and A. Kalai, "A note on learning from multiple-instance examples," *Machine learning*, vol. 30, no. 1, pp. 23–29, 1998.

[33] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[34] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[35] A. Tharwat and W. Schenck, "A survey on active learning: state-of-the-art, practical challenges and research directions," *Mathematics*, vol. 11, no. 4, p. 820, 2023.

[36] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 1321–1330.

[37] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[38] A. Kirsch, J. van Amersfoort, and Y. Gal, "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[39] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach*. Mit Press, 2022.

[40] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 927–946.

[41] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1462–1469, 2019.

[42] B. Peng, Y. Mu, Y. Guan, S. E. Li, Y. Yin, and J. Chen, "Model-based actor-critic with chance constraint for stochastic system," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 4694–4700.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[44] N. Fang, G. Liu, and W. Gong, "Offline inverse constrained reinforcement learning for safe-critical decision making in healthcare," *IEEE Transactions on Artificial Intelligence*, vol. 7, no. 4, pp. 2037–2046, 2026.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on October 26,2026 at 09:32:22 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.