2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Model-Based Runtime Monitoring with Interactive Imitation Learning

Huihan Liu[1], Shivin Dass[1], Roberto Martín-Martín[1], Yuke Zhu[1]

***Abstract*— Robot learning methods have recently made great strides, but generalization and robustness challenges still hinder their widespread deployment. Failing to detect and address potential failures renders state-of-the-art learning systems not combat-ready for high-stakes tasks. Recent advances in interactive imitation learning have presented a promising framework for human-robot teaming, enabling the robots to operate safely and continually improve their performances over long-term deployments. Nonetheless, existing methods typically require constant human supervision and preemptive feedback, limiting their practicality in realistic domains. This work aims to endow a robot with the ability to monitor and detect errors during task execution. We introduce a model-based runtime monitoring algorithm that learns from deployment data to detect system anomalies and anticipate failures. Unlike prior work that cannot foresee future failures or requires failure experiences for training, our method learns a latent-space dynamics model and a failure classifier, enabling our method to simulate future action outcomes and detect out-of-distribution and high-risk states preemptively. We train our method within an interactive imitation learning framework, where it continually updates the model from the experiences of the human-robot team collected using trustworthy deployments. Consequently, our method reduces the human workload needed over time while ensuring reliable task execution. Our method outperforms the baselines across system-level and unit-test metrics, with 23% and 40% higher success rates in simulation and on physical hardware, respectively. More information at** `https://ut-austin-rpl.github.io/sirius-runtime-monitor/`

I. Introduction

We have witnessed tremendous progress in learning-based robotics systems in recent years [1, 4, 24]. Despite exciting showcases in research settings, they continue to struggle with generalization and reliability issues for widespread deployment. To achieve continual model improvement in trustworthy deployments, a burgeoning body of work [28, 29] has explored the shift from the conventional "train-then-deploy" paradigm to "learning on the job" with human-robot teams. Particularly, interactive imitation learning (IIL) [6] has advocated a framework where the human supervises the robot's execution and performs interventions to handle difficult situations, and the robot continually learns from deployment data. Existing IIL approaches typically require humans to continuously monitor the system and provide imminent feedback, incurring prohibitive human workloads in realistic applications. A critical step toward making these methods practical is incorporating a *runtime monitoring* mechanism, allowing the robots to self-monitor and predict errors during task execution [50].

Runtime monitoring mechanisms have been investigated in two main directions: unsupervised *out-of-distribution (OOD) detection* and supervised *failure detection*. Unsupervised out-of-distribution (OOD) detection methods [8, 31, 34, 45] use the data distributions as a proxy to estimate errors. They often struggle to faithfully capture the error patterns and may overlook critical situations [36, 47]. More critically, these methods determine whether a failure occurs based on the current system states but cannot identify future failures before they become unavoidable. Supervised approaches train failure detection models from actual failure experiences. Failure detection can be done by classifying normal and failure examples [12, 48] or estimating risk Q-values to predict future failures [18, 19]. These approaches are more accurate in capturing error patterns and preempting failures. Nonetheless, they heavily rely on explicit failures encountered in the past as supervisory signals. In deployed systems, failures can be catastrophic, and acquiring such failure data hinders the system's safety and reliability.

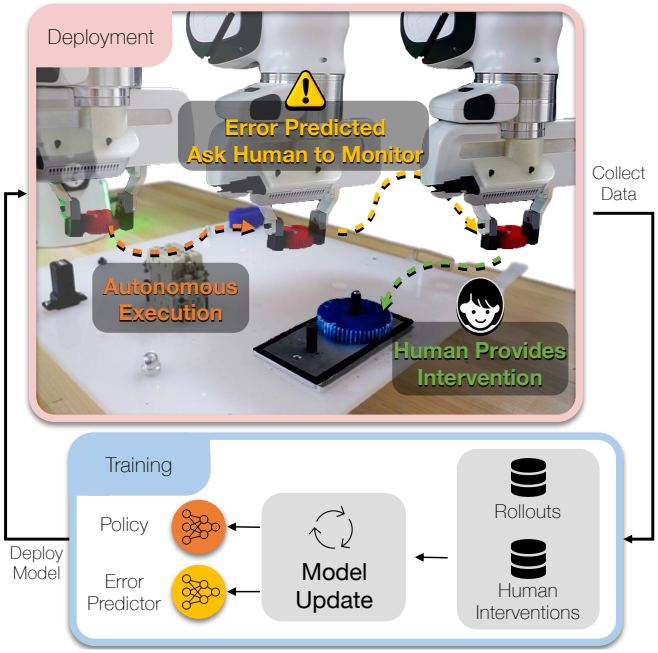

Fig. 1: **Overview.** We introduce a model-based runtime monitoring algorithm that continuously learns to predict errors from deployment data. We integrate this runtime monitoring algorithm into an interactive imitation learning framework to ensure trustworthy long-term deployment.

To this end, we introduce a runtime monitoring algorithm as an extension to Sirius [28], our prior work that improves robot autonomy over long-term deployments with humans in the loop. The runtime monitoring algorithm proactively detects system anomalies and preemptive failures in the interactive imitation learning framework of Sirius (see Fig. 1). Our design follows two conceptual ideas: First, it adopts

[1]The University of Texas at Austin. Correspondence: huihanl@utexas.edu

a *model-based* approach, which trains a predictive model of the environment dynamics for failure prediction. The dynamics model can simulate future policy rollouts and predict upcoming failures. Second, it uses an *intervention-informed* approach to train its model components over long-term deployments. The algorithm harnesses the inherent structure of human interventions to continually learn an error predictor without having to encounter explicit failures, thus ensuring trustworthy task execution.

Our method performs model-based runtime monitoring with two learnable components: a *dynamics model* and a *failure classifier*. We first construct a latent space, where image observations are encoded into feature vectors as the latent states. We train a dynamics model that predicts the next latent state conditioned on the current observation and the action. We also train a policy from the same latent space. The latent state space shared between the dynamics model and the policy allows our method to simulate counterfactual trajectories and predict different action outcomes [46]. We also train a failure classifier that predicts whether a future state leads to failure. With these two components, an error is identified by out-of-distribution (OOD) detection with the dynamics model and failure detection with the dynamics model and the failure classifier. Contrary to prior work [18, 34] that uses isolated OOD and failure detection systems, we find it effective to unify them in a single model, enhancing the data efficiency and overall performance of our system.

Prior learning-based error predictors [11, 18, 48] typically require explicit failures to form training datasets, which are infeasible to collect during safe deployments. Instead, we leverage human interventions to inform the error predictor of the moments when humans perceive the system is at risk of failure. Human interventions offer implicit cues of their judgment on whether the robot's actions may be incorrect, undesirable, or close to failing [18, 28, 29]. We train the failure classifier with human interventions and robot rollouts collected from deployments and continuously update it over time. As a result, the failure classifier can dynamically adapt to the policy's changing behaviors.

We tested our method on two simulated and two real-world tasks with a human-in-the-loop learning and deployment framework [18, 28]. Our method achieves on average 32% higher success rates and 16% more efficient use of human workload utilization than state-of-the-art interactive imitation learning baselines. In summary, our main contributions are as follows:

- We introduce a model-based runtime monitoring algorithm that proactively solicits human help based on future error preemption;
- We develop an effective learning method to jointly and continually update our error predictor and policy during human-in-the-loop deployments;
- We systematically evaluate our method against baselines in simulation and on physical hardware and demonstrate its effectiveness at the system level and in unit tests.

II. Related Work

Interactive Imitation Learning. In interactive imitation learning [2], robots receive human feedback during task execution, allowing for continuous improvements of the policy performances [6]. The human involvement in the learning loop has two ways: 1) *human-gated*, where the human constantly supervises the robot and decides when to provide feedback [25, 28, 35]; or 2) *robot-gated*, where the robot actively solicits human feedback [18, 20, 31]. The robot-gated decision is typically based on heuristic metrics such as ensemble variance [31], risk [18], or task uncertainty [8]. While prior robot-gated works have studied efficiently sourcing human feedback for learning, this work investigates automated error prediction with the goal of minimizing runtime errors and ensuring trustworthy deployment.

Runtime Monitoring in Robot Learning. Runtime monitoring and error prediction methods have received considerable interest in robotics [21, 39, 49]. For robot learning methods, in particular, there have mainly been two bodies of work: out-of-distribution (OOD) detection [36, 40], and failure detection [10]. Unsupervised out-of-distribution (OOD) detection methods [8, 31, 34, 45] use the degree of OOD as an approximate proxy for error prediction. The supervised failure detection approach performs binary classification of normal/failure states [12, 48] with a classifier trained on success and failure examples. A recent line of work [18, 19] learns a risk Q function from agent rollouts to identify states that are likely to lead to failures using techniques developed in safe reinforcement learning (RL) [5, 41, 42].

Model-based Learning Approach. Learned dynamics models have been adopted in reinforcement learning (RL) [13–15, 23, 46], imitation learning [9, 22], planning [3, 7, 16, 37]. They have also demonstrated success in different robotics applications [32, 33, 38, 46]. Prior work uses dynamics learning for safe RL to learn a safe policy given known safety violation criteria [43]. In the same vein, we harness the predictive ability of model-based approaches for runtime monitoring. Concretely, we use the dynamics model to predict the outcomes of future policy actions in a latent space [27], allowing us to identify risky states *before* they occur and ask for timely human supervision.

III. Model-based Runtime Monitoring

We start by describing our problem formulation in Section III-A. We then discuss our algorithmic components in Section III-B, and how the individual modules (Section III-C) and the overall system (Section III-D) are used for runtime monitoring in practice.

A. Problem Formulation

We formulate a robot manipulation task as a discrete-time Markov Decision Process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, with continuous states $s \in \mathcal{S}$, continuous actions $a \in \mathcal{A}$, unknown transition dynamics $\mathcal{P}(.|s, a)$, reward function $\mathcal{R}(s, a, s')$ and discount factor $\gamma \in [0, 1)$. We aim to learn a task-oriented robot policy $\pi_r : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes the

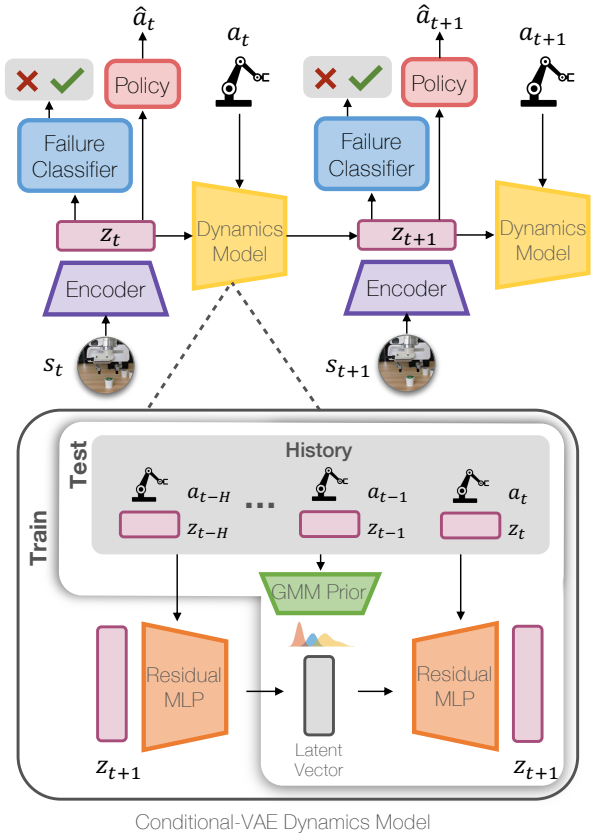

Fig. 2: **Model Architecture.** We train a dynamics model, implemented as a conditional Variational Autoencoder (cVAE), to predict the next latent state given the current state and action. We also train a policy and a failure classifier head based on the latent state. The dynamics model and policy are trained from the experiences collected from task execution. The failure classifier uses the human intervention states to infer failure states.

return $\mathbb{E}[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t, s_{t+1})]$ while minimizing the number of failures during its deployment.

We consider a human-in-the-loop learning and deployment framework, where a robot performs tasks with humans available to provide corrective feedback in the form of interventions. In contrast to prior work [26, 28, 29] where the human continuously monitors the system and provides feedback whenever necessary, our work focuses on developing a runtime monitoring mechanism that actively solicits human feedback only when an error is detected.

B. Model-based Method Design

Here we discuss our model-based approach to runtime monitoring. As shown in Figure 2, our approach comprises of four main components denoted as $\mathcal{W} = (E_\gamma, T_\psi, \pi_\theta, C_\lambda)$. The observation encoder E_γ translates raw image and robot proprioceptive state observations into latent state embeddings. These embeddings form a shared latent space wherein the dynamics model T_ψ, the policy π_θ, and the failure classifier C_λ all operate.

Dynamics Model and Policy. The observation encoder E_γ first encodes the raw observation into a latent state space representation. The dynamics model T_ψ predicts future latent state rather than future raw observations, enabling fast sampling of futures as noted in prior work [13, 15, 46]. Given the input sequence of past latent states' embeddings and the current action, the model outputs the next latent state. The dynamics model uses a conditional Variational Autoencoder (cVAE). The stochastic latent space of cVAE supports the sampling of multiple future predictions, helping to model future outcomes accurately.

The policy π_θ is jointly trained with the dynamics model and shares the same latent embedding space so as to enable policy rollouts in the latent space. The overall training objective combines the maximum likelihood behavior cloning (BC) objective with the VAE loss terms, which include Kullback-Leibler (KL) divergence and reconstruction loss of the next state latent embedding:

$$L_W = \mathop{\mathbb{E}}_{\substack{(s,a,s')\sim\mathcal{D} \\ q(z|s',s,a)}} - \log \pi_\theta(a \mid s) - \log p(s'|s,a,z) + D_{KL}(q(z|s',s,a)||p(z|s,a)) \quad (1)$$

Failure Classifier. After jointly training the policy and dynamics model, we train a failure classifier C_λ on the frozen latent space. Our failure labels come from moments in past interaction data when human intervenes. Based on prior work [28], we observe that pre-intervention points, *i.e.*, the states right before each human intervention, reflect the human's real-time judgment of undesirable or "failure" states. Therefore, these labels act as indicators of human risk assessment and can be used for training the failure classifier.

One particular design choice is that rather than a binary classification of failure versus normal states, we adopt a three-class classification of failure state, normal rollout, and intervention state. This is because we observe that human interventions and system failures often occur in similar states. Therefore, separating these two categories provides a more informative learning signal and improves the failure classifier accuracy. We train this classifier with a cross-entropy loss objective $L_C = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$ with balanced sampling.

Implementation Details. We implement our behavior cloning policy with a backbone of BC-RNN using ResNet encoders [17]. We implement the cVAE dynamics model with GMM latent vector [30] and residual MLP [44] encoder and decoder (see Figure 2 bottom), making both the latent vector and decoder conditioned on the state and action input. We train the failure classifier with an LSTM, which takes in the history of states and actions and outputs the probability of failure.

C. Runtime Monitor in Operation: Modules

During deployment, the system draws samples from the cVAE dynamics model to construct N future policy rollouts in the latent space. This process allows us to predict l steps ahead by iteratively running the dynamics model and the policy. Since the cVAE model has a stochastic latent space to draw samples from, we can generate multiple predictions of future states. We evaluate each future independently and then average the error predictor results. Each predicted future state undergoes a two-level evaluation, which assesses two aspects of the system:

Modules	Hyperparameter	Value
Dynamics Model (shared)	Number of futures, N	200
	Future horizon length, l	10
OOD Detection	Nearest-neighbor threshold, α	0.06
Failure Detection	Future % threshold, β	0.6
	Number of failures in history, K	5
	History length, P	10

TABLE I: Hyperparameter choices for our method.

Out-of-Distribution (OOD) Detection. We evaluate whether a state is OOD by calculating the nearest neighbor distance to the latent embeddings (obtained from E_γ) of the in-distribution data, *i.e.*, expert demonstrations collected by humans. We measure the mean distance of multiple imaginary futures and classify a state as OOD if the nearest neighbor distance is above a threshold α.

Failure Detection. If the state is not classified as OOD, Our method anticipates whether there will be a future failure. We employ the failure classifier C_λ which classifies a state as a failure when the averaged failure classification result across multiple future rollouts exceeds a threshold β, indicating a high probability that this state will result in a failure. Human supervision is only solicited if K of the past P states are identified as failures, ensuring temporally consistent predictions and reducing the false positive rates.

Overall, the model-based future prediction enables effective preemptive future state forecasting, and the complementary roles of OOD detection and failure detection ensure reliable real-time monitoring during operation. The detailed hyperparameter choices are listed in Table I.

D. Runtime Monitoring in Operation: System

We now elaborate on the procedure for online data aggregation and training. We initialize a base model $\mathcal{W}^0 = (E^0_\gamma, \pi^0_\theta, T^0_\psi, C^0_\lambda)$ using the initial human demonstration dataset $\mathcal{D}^0$. In every subsequent round r of deployment, the robot executes the policy π^r_θ, while our error predictor performs runtime monitoring. If our method detects OOD or failure, it requests human monitoring. Note that in round 1, our error predictor asks for full human monitoring to obtain the human intervention labels that bootstrap the training of the failure classifier. While monitoring, the humans may choose to *actively intervene* if they see undesirable states from their judgment. We denote the points of *active intervention* as the intervention label $h_t \in \{0, 1\}$, an indicator variable that is 1 if the human supervisor actively intervenes at that step. This is what is used to train the failure classifier C^r_λ. Each of the transitions (s_t, a_t, s_{t+1}, h_t) are then stored in a new dataset $\mathcal{D}^r$. For the next round $r+1$, we retrain our models on the combined dataset $\bigcup_{i=0}^{r} D^i$ to obtain $\mathcal{W}^{r+1}$.

IV. Experiments

In our experiments, we seek to answer the following questions: 1) How effective is our method at ensuring trustworthy system performance? 2) Can our method utilize the human supervision input effectively? 3) Quantitatively, how accurate is our method at predicting errors?

Tasks	Methods	Round 1	Round 2	Round 3
Nut Assembly	**Ours**	–	**98.5**	**100.0**
	ThriftyDAgger	86.6	90.0	88.1
	PATO	82.0	80.9	87.5
	MoMaRT	56.0	60.0	70.0
Threading	**Ours**	–	**93.2**	**98.7**
	ThriftyDAgger	79.5	75.6	77.5
	PATO	32.0	82.9	78.1
	MoMaRT	34.0	76.0	80.0
Coffee Pod Packing (real)	**Ours**	–	**83.5**	**92.5**
	ThriftyDAgger	23.1	50.9	57.5
Gear Assembly (real)	**Ours**	–	**80.0**	**82.9**
	ThriftyDAgger	44.0	62.0	70.0

TABLE II: **Combined Policy Performance (in Success Rate).** Our method consistently outperforms the baseline over the rounds. Note that the Round 1 results of Ours are N/A as it uses full human monitoring for warm-start.

To answer these questions, we conduct two sets of experiments 1) to evaluate the overall *system-level performance*; 2) to *unit test* the performance of error predictors, each of which we elaborate on in the subsequent sections.

A. System-Level Performance

We evaluate the performance of a human-in-the-loop system in an iterative deployment loop. The robot is initially bootstrapped from a BC policy and continues to execute a task with runtime monitoring. The deployment data are aggregated for training the policy to be deployed next round. We run the deployment loop over three repeated rounds, following prior setup [26, 28, 29]. Each round consists of 1) data aggregation with the trained policy and human interventions facilitated by the learned error predictors and 2) training the model over all previously collected data, as described in Section III-D. To ensure a fair comparison in the system-level evaluations, we maintain a consistent number of interventions across our approach and the baseline methods.

1) Evaluation Protocol: To benchmark the system-level performance of the runtime monitoring system, we would like to evaluate two aspects:

- **Collaborative policy performance** measures the overall performance of the system when actively asking for human supervision using the error predictor. It assesses how reliable the runtime monitoring is in guarding the system from failures and facilitating successful task completions. We measure this in the overall task success rate.
- **Optimized use of human input** measures how the system leverages human input to maximize task outcomes in a fixed time frame. Intuitively, we want the system to engage human assistance precisely when it is most necessary. To quantify this, we adapt the Return On Human Effort (ROHE) from [20] for a single robot case and normalize it to capture success rates,

$$\text{Normalized ROHE} = \frac{\mathbb{E}_\tau[\sum_{t=0}^{T_\tau} r_t^\tau]}{1 + \frac{H}{T}} \tag{2}$$

where H is the total number of human interventions across trajectories, T is the total number of time steps, and r_t^τ is the reward obtained from performing trajectory τ. In

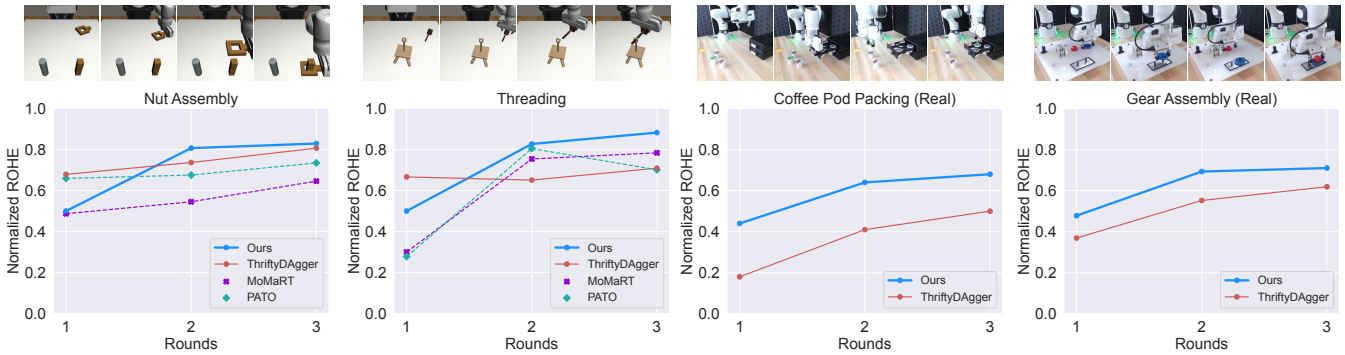

Fig. 3: Normalized ROHE curves over three rounds of iterative deployment for tasks Square Nut Assembly (left), Threading (left-middle), Coffee Pod Packing (right-middle), and Gear Assembly (right). Our method generally has lower ROHE in the first round due to the higher human engagement initially; the ROHE becomes better in later rounds as our method becomes more effective at identifying important errors during deployment.

Task	Metrics	**Ours**	ThriftyDAgger	MoMaRT	PATO
Nut Assembly	IOU ↑	**0.279**	0.153	0.074	0.089
	DCI ↓	**89.1**	94.7	172.7	112.1
Threading	IOU ↑	**0.317**	0.044	0.077	0.078
	DCI ↓	**30.1**	60.7	129.8	47.1
Coffee Pod	IOU ↑	**0.205**	0.089	0.103	0.101
	DCI ↓	**37.7**	76.5	83.6	47.9
Gear Assembly	IOU ↑	**0.255**	0.197	0.151	0.120
	DCI ↓	**23.5**	32.2	52.7	27.9

TABLE III: **Unit testing error predictors.** Our method outperforms other baselines in the two metrics. Better IOU performance indicates higher overlap between detected and human-labeled failures, and lower DCI means that our method's failure events are closer to the true human failure labeling.

sparse reward settings, the numerator is the success rate of the policy while the term in the denominator penalizes the model from asking for excessive human interventions.

2) Baselines: For system-level comparison, we compare our method with **ThriftyDAgger** [18], a state-of-the-art interactive imitation learning baseline that queries human correction in continuous policy updates. ThriftyDAgger detects novel states by training an ensemble of policies and measuring the variance between their outputs. The riskiness of a state is estimated by learning a Q-function from the data and policy rollouts. Note that the additional step of policy rollouts is required for collecting failure data to train the Q-function, which could be infeasible in safety critical domains.

We also compare to two other baselines that were originally not designed for continuous policy update: **MoMaRT** [45] and **PATO** [8]. MoMaRT uses VAE reconstruction error for the input images as an indicator for out-of-distribution states. PATO uses variance from ensemble policy to measure uncertainty similar to [18], as well as the variance of future latent state prediction from VAE to measure the uncertainty of the task. We run these two baselines on the deployment data generated by our method for a fair comparison of the two simulation tasks.

3) Results: We perform two contact-rich manipulation tasks, **Nut Assembly** and **Threading**, in the robosuite simulator [51], and **Coffee Pod Packing** and **Gear Assembly** in the real world. For all tasks, we use a Franka Emika Panda robot arm equipped with a parallel jaw gripper, and the operational space controller (OSC).

We evaluate the collaborative policy performance and the Normalized ROHE score defined in Section IV-A.1 at each deployment round. Our method outperforms baselines in terms of collaborative policy performance (see Table II), and Normalized ROHE score (see Figure 3). We attribute the higher performance of our method to its model-based design: the dynamics model enables future latent state prediction that calls for more timely interventions, resulting in better system performance. Note that to train our method's failure classifier, we perform the initial round (Round 1) of deployment with full human monitoring to obtain the human intervention labels. This is reflected in a lower Normalized ROHE score in Round 1 as seen in Figure 3. It, in turn, allows our method to model the failure states well in the later rounds, resulting in a higher combined success rate (see Table II) and improved ROHE score (see Round 2 and 3 in Figure 3).

In contrast, **ThriftyDAgger** and **PATO** employ ensemble policies for OOD detection, which reflects the task uncertainty (multimodality) in the original data, *e.g.*, when the policy is taking gripper actions such as grasping and releasing, resulting in many false positives. **ThriftyDAgger** also uses a risk Q function to measure the probability of failure. We observe that the values of the trained Q function tend to increase as the robot makes task progress, leading to a higher number of false positives at the beginning of a task and false negatives towards the end, for a fixed threshold value. Meanwhile, **MoMaRT** exhibits some known issues of over-generalization presented in reconstruction-based methods [36, 47]. Specifically, for the fine-grained robot manipulation tasks we examine in this work, failure or success in robot manipulation (e.g. when inserting into a tight hole) might only result in small image variations. VAE reconstruction error often fails to capture such minor pixel differences, leading to numerous false negatives.

B. Unit Testing Error Predictors

For a more systematic study of error predictor performance, we examine how accurately error predictors capture the actual human risk assessment. We unit test the error predictors' accuracy using *active* human monitoring: we deploy a learned policy within an environment in which a human operator fully supervises the system and can intervene

whenever an unsafe state is observed [28, 29]. This method enables us to obtain the set of failure states as determined by human observers. We then apply the learned error predictors to the collected trajectories. For each trajectory, we obtain the human intervention labels as well as the predicted failure labels. We then compare how close the failure predictions of the error predictors are to that of the human operators. It enables us to quantitatively measure the performance of the learned error predictors independent of the system deployment loop.

We use two metrics to evaluate the error predictors:

- **IOU (Intersection over Union)**: we calculate the intersection over union between the predicted failure regions of the error predictor and that of a human. We define the intersection and union for our 1D case as follows:
  - *Intersection*: Number of timesteps that belong to the predicted failure regions of both the human AND the error predictor.
  - *Union*: Number of timesteps that belong to the predicted failure regions of either the human OR the error predictor. Intuitively, if the error predictor predicts the failures accurately, the number of timesteps belonging to the intersection would be larger, leading to a higher IOU.
- **DCI (Distance to Closest Intervention)**: for each human intervention, we measure the distance to the closest predicted failure. For each predicted failure, we measure the distance to the human intervention. Finally, we take their average. DCI penalizes predicted failures that are temporally further away from true interventions.

The results in Table III show that our method can accurately predict the intervention points by humans and perform better than the baselines on all tasks. This is mostly attributed to our method's intervention-informed approach. Unlike other methods that uses metrics unrelated to the risk assessment by humans, our method incorporates human intervention data as part of the learning process. By treating human interventions as supervisory signals, our method adaptively learns from the decision-making process of human operators that other metrics struggle to capture. It enables our method to identify potential failures more aligned with human risk assessment than other methods.

C. Ablation Study

We further evaluate the individual effects of our method's OOD detection and failure detection module on runtime monitoring performance. We study this in two simulation tasks, Nut Assembly and Threading. We use the Round 3 setting from Table II with the same policy and test the collaborative policy performance of our method using OOD detection only and failure detection only, denoted as Ours OOD and Ours Failure, respectively. Figure 4 shows that the combined performance of Ours (Full) outperforms the individual modules, Ours Failure and Ours OOD. We hypothesize that this is because the OOD detection and failure detection modules perform complementary functions. When the robot operates in novel scenarios, OOD detection plays a role in identifying unfamiliar states that may not necessarily be immediate failures but could lead to failures if not addressed. Meanwhile, the failure detection module is capable of identifying subtle changes in critical bottleneck states that OOD detection may overlook. We show example situations where OOD detection and failure detection take effect in Figure 5. The two modules work in coordination to identify different kinds of errors, achieving reliable runtime monitoring for continual robot deployment.

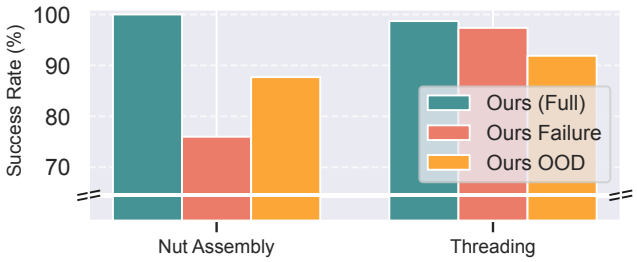

Fig. 4: **Ablation Study.** Ours (Full) achieves a higher overall performance than Ours Failure and Ours OOD. This result indicates that both failure detection and OOD detection modules complement each other and contribute to the overall success of our system.

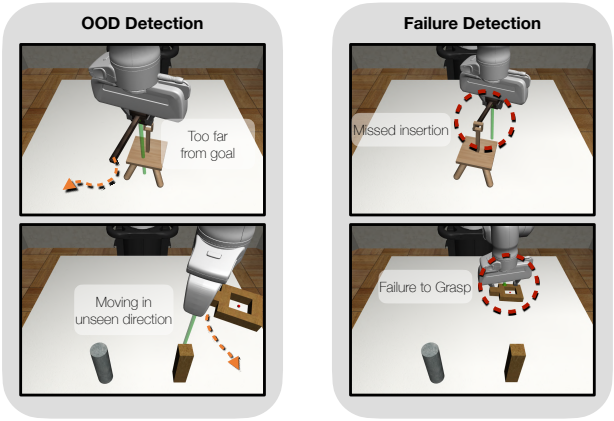

Fig. 5: **Qualitative results from ablation study.** The OOD detection is able to identify unfamiliar states at a coarser level while the failure detection identifies finer-grained failures.

V. Conclusion and Limitations

We present a runtime-monitoring algorithm that uses deployment data to predict errors and seek human assistance. We integrate this algorithm into a human-in-the-loop interactive imitation learning system. Results showcase that our method is effective for ensuring trustworthy deployment and facilitating policy learning over long-term deployments.

Our experiments, conducted by a single human subject per task, may not faithfully reflect the representative performance of human-robot teams across a broader population. A large-scale study on human intervention behavior and its impact on our method's ability to capture human-guided failure modes would provide valuable insights. Furthermore, our work has focused on quasi-static manipulation tasks, where mistakes are relatively easy to recover from through teleoperation. We are interested in exploring extending our system's applicability to more dynamic environments, diverse robot platforms beyond robot arms, and more complex tasks.

ACKNOWLEDGMENT

We thank Ajay Mandlekar for sharing well-designed simulation task environments. We thank Soroush Nasiriany, Yifeng Zhu, Mingyo Seo, Jake Grigsby, Quantao Yang and Yue Wu for providing helpful feedback for this manuscript. We thank Yue Zhao, Bo Liu and Zhenyu Jiang for their fruitful discussions. We acknowledge the support of National Science Foundation (2145283, 2318065), the Office of Naval Research (N00014-22-1-2204), Amazon, and UT Good Systems.

REFERENCES

[1] M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," vol. 39, 2018, pp. 20–3.

[2] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics Auton. Syst.*, vol. 57, pp. 469–483, 2009.

[3] A. Argenson and G. Dulac-Arnold, "Model-based offline planning," *arXiv preprint arXiv:2008.05556*, 2020.

[4] A. Brohan *et al.*, *Rt-1: Robotics transformer for real-world control at scale*, 2022. arXiv: 2212.06817 [cs.RO].

[5] L. Brunke *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *ArXiv*, vol. abs/2108.06266, 2021.

[6] C. Celemin *et al.*, *Interactive imitation learning in robotics: A survey*, 2022. arXiv: 2211.00600 [cs.RO].

[7] M. H. Danesh, P. Cai, and D. Hsu, "Leader: Learning attention over driving behaviors for planning under uncertainty," *ArXiv*, vol. abs/2209.11422, 2022.

[8] S. Dass, K. Pertsch, H. Zhang, Y. Lee, J. J. Lim, and S. Nikolaidis, "Pato: Policy assisted teleoperation for scalable robot data collection," *arXiv preprint arXiv:2212.04708*, 2022.

[9] B. DeMoss, P. Duckworth, N. Hawes, and I. Posner, "Ditto: Offline imitation learning with world models," *arXiv preprint arXiv:2302.03086*, 2023.

[10] A. Diryag, M. Mitić, and Z. Miljković, "Neural networks for prediction of robot failures," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 228, no. 8, pp. 1444–1458, 2014.

[11] C. Gokmen, D. Ho, and M. Khansari, "Asking for help: Failure prediction in behavioral cloning through value approximation," *arXiv preprint arXiv:2302.04334*, 2023.

[12] C. Gokmen, D. Ho, and M. Khansari, "Asking for help: Failure prediction in behavioral cloning through value approximation," *ArXiv*, vol. abs/2302.04334, 2023.

[13] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[14] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.

[15] N. Hansen, Y. Lin, H. Su, X. Wang, V. Kumar, and A. Rajeswaran, "Modem: Accelerating visual model-based reinforcement learning with demonstrations," *arXiv preprint arXiv:2212.05698*, 2022.

[16] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," *arXiv preprint arXiv:2203.04955*, 2022.

[17] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[18] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, "Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning," *arXiv preprint arXiv:2109.08273*, 2021.

[19] R. Hoque *et al.*, *Fleet-dagger: Interactive robot fleet learning with scalable human supervision*, 2022. arXiv: 2206.14349 [cs.RO].

[20] R. Hoque *et al.*, "Fleet-dagger: Interactive robot fleet learning with scalable human supervision," in *Conference on Robot Learning*, PMLR, 2023, pp. 368–380.

[21] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," 2023.

[22] A. Hu *et al.*, *Model-based imitation learning for urban driving*, 2022. arXiv: 2210.07729 [cs.CV].

[23] M. Janner, J. Fu, M. Zhang, and S. Levine, *When to trust your model: Model-based policy optimization*, 2019. arXiv: 1906.08253 [cs.LG].

[24] D. Kalashnikov *et al.*, "QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation," in *CoRL*, 2018.

[25] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8077–8083.

[26] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," *2019 International Conference on Robotics and Automation (ICRA)*, May 2019.

[27] L. Lehnert and M. L. Littman, "Successor features support model-based and model-free reinforcement learning," *CoRR*, vol. abs/1901.11437, 2019. arXiv: 1901.11437.

[28] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu, "Robot learning on the job: Human-in-the-loop autonomy and learning during deployment," *arXiv preprint arXiv:2211.08416*, 2022.

[29] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *arXiv preprint arXiv:2012.06733*, 2020.

[30] A. Mandlekar *et al.*, *What matters in learning from offline human demonstrations for robot manipulation*, 2021. arXiv: 2108.03298 [cs.RO].

[31] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "EnsembleDagger: A bayesian approach to safe imitation learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 5041–5048.

[32] R. Mendonca, S. Bahl, and D. Pathak, *Alan: Autonomously exploring robotic agents in the real world*, 2023. arXiv: 2302.06604 [cs.RO].

[33] R. Mendonca, S. Bahl, and D. Pathak, *Structured world models from human videos*, 2023. arXiv: 2308.10901 [cs.RO].

[34] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," *Robotics: Science and Systems*, Jul. 2017.

[35] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[36] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou, "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges," *CoRR*, vol. abs/2110.14051, 2021. arXiv: 2110.14051.

[37] I. Schubert *et al.*, "A generalist dynamics model for control," *arXiv preprint arXiv:2305.10912*, 2023.

[38] L. X. Shi, J. J. Lim, and Y. Lee, "Skill-based model-based reinforcement learning," *arXiv preprint arXiv:2207.07560*, 2022.

[39] R. Sinha, E. Schmerling, and M. Pavone, "Closing the loop on runtime monitors with fallback-safe mpc," in *Proc. IEEE Conf. on Decision and Control*, Submitted, 2023.

[40] R. Sinha *et al.*, *A system-level view on out-of-distribution data in robotics*, 2022. arXiv: 2212.14020 [cs.RO].

[41] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep rl with a safety critic," *arXiv preprint arXiv:2010.14603*, 2020.

[42] B. Thananjeyan *et al.*, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[43] G. Thomas, Y. Luo, and T. Ma, *Safe reinforcement learning by imagining the near future*, 2022. arXiv: 2202.07789 [cs.LG].

[44] Z. Wang *et al.*, *Critic regularized regression*, 2020. arXiv: 2006.15134 [cs.LG].

[45] J. Wong *et al.*, "Error-aware imitation learning from teleoperation data for mobile manipulation," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 1367–1378.

[46] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on Robot Learning*, PMLR, 2023, pp. 2226–2240.

[47] Z. Xiao, Q. Yan, and Y. Amit, "Likelihood regret: An out-of-distribution detection score for variational auto-encoder," *CoRR*, vol. abs/2003.02977, 2020. arXiv: 2003.02977.

[48] A. Xie, F. Tajwar, A. Sharma, and C. Finn, *When to ask for help: Proactive interventions in autonomous reinforcement learning*, 2022. arXiv: 2210.10765 [cs.LG].

[49] E. Yel and N. Bezzo, "Fast run-time monitoring, replanning, and recovery for safe autonomous system operations," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1661–1667.

[50] H. Yoon and S. Sankaranarayanan, "Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021.

[51] Y. Zhu *et al.*, *Robosuite: A modular simulation framework and benchmark for robot learning*, 2020. arXiv: 2009.12293 [cs.RO].