Multi-Task Interactive Robot Fleet Learning with Visual World Models

Anonymous Author(s)
Affiliation
Address
email

Abstract: Recent advancements in large multi-task robot agents hold promise of robot fleets operating in household and industrial settings, capable of performing diverse tasks across various environments. Despite these advances, these robots often struggle with generalization and robustness in real-world, unstructured settings, limiting their practical deployment. To address these issues, we propose FIREWORK, a framework for multi-task interactive robot fleet learning, supervising the continual policy update throughout deployment with efficient multi-task runtime monitoring and humans in the loop. We realize this with a visual world model to predict future task outcomes, which serves as the backbone for learned representation across downstream error prediction tasks. We train dynamic error predictors that auto-adjust their prediction criterias to adapt to the robot's evolving autonomy, reducing human workload over time. Evaluations on large-scale benchmarks demonstrate our framework's effectiveness in improving multi-task policy performance and monitoring accuracy. We demonstrate the performance of FIREWORK on robocasa in simulation and mutex in real world, two large, diverse multi-task environment benchmarks. More information at project website https://firework-corl2024.github.io/

Keywords: Robot Manipulation, Interactive Imitation Learning, Fleet Learning

1 Introduction

In recent years, great progress has been made in developing generally capable robot agents for performing a wide range of tasks [1, 2, 3]. The rapid progress in generalist robots holds promise for a future of robot fleets [4, 5, 6] operating in households and industrial settings on a generalist multi-task policy. Despite the progress in research development, the robots can still fall short of generalization and robustness when being deployed in the real world, where environments are diverse and unstructured; this undermines the safety and reliability of the system and limits the model's applicability to real-world deployment.

To mitigate these challenges, numerous works in the field of interactive imitation learning (IIL) [7, 8, 9, 10, 11, 12, 13] and interactive fleet learning (IFL) [14, 5, 6] have been proposed. Prior works in human-in-the-loop learning [7, 13, 15, 10] have proposed to use human monitoring and human correction on the job to ensure trustworthy deployment; however, they require constant human monitoring and incur high human workload. To reduce the amount of human workload, the runtime monitoring approaches [16, 17, 18, 19, 12, 20, 21] propose using error detectors to automatically monitor robot performance, detect system anomalies and ask for human control when needed. Prior works has proposed out-of-distribution (OOD) detection [20, 21, 18] or failure detection [12] methods to detect error cases of robot execution. However, these works have primarily been applied for task-specific policies within a controlled single-task setting, limiting the efficiency and effectiveness in the multi-task fleet deployment.

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

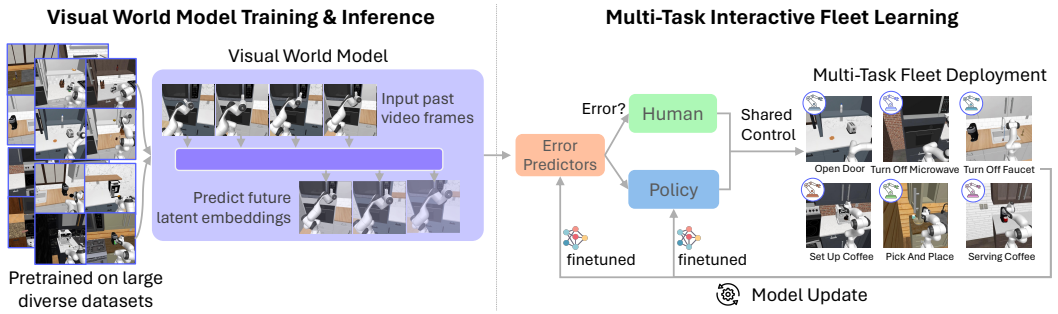

Figure 1: **Multi-Task Fleet Learning.** We first train a visual world model and then use it for multi-task fleet learning and deployment.

We introduce FIREWORK[1], a framework for multi-task interactive robot fleet learning, where multi-task policies can be deployed to massive robot fleets with humans in the loop and efficient runtime monitoring. The key challenges are 1) to enable a multi-task policy to be continually updated throughout deployment with the diverse, massive-scale deployment data, enabling sharing of knowledge across tasks; 2) to build a runtime monitoring mechanism which efficiently monitor different tasks simultaneously across different diverse scenes throughout the deployment; To tackle the first challenge, we train a multi-task policy that is able to be constantly finetuned on deployment data learning from the data on the job; To tackle the second challenge, we train a visual world model as the backbone of the runtime monitoring, whose learned representation can be shared and used across downstream error predictors for various tasks.

Recent advancement of world models [22, 23, 24, 25] enables simulation into the future, predicting future progress outcomes. Inspired by such progress, we build a visual world model that is trained on large diverse experience of robot trajectory frames. This enables future prediction of robot task progress, allowing us to see future outcome and preempt potential failures from happening. The learned representation from the model are then and shared in downstream error prediction across different tasks. Building on the frozen representation from the world model, we train error predictors based on failure and OOD detection which can are continuously finetuned throughout deployment. Rather than having a fixed detection threshold as in prior works, we auto-adjust the error predictor's prediction criteria based on the current task performance using the human interaction trace as a proxy. We find that the auto threshold adjustment can customize to the evolving level of robot autonomy, giving better runtime monitoring results.

We test our multi-task interactive robot fleet learning framework on large multi-task benchmarks both in simulation and on real robots. Our key findings are 1) our runtime monitoring can effectively supervise diverse multi-task scenarios with high performances; 2) the multi-task policy is able to continually improve over time with the data collected at fleet deployment; 3) our error predictors for runtime monitoring surpass baselines in their accuracy and use of human workload. In summary, our contributions are as follows:

1. A framework of multi-task interactive robot fleet learning where the multi-task robot policy efficiently improves over deployment with runtime monitoring and human in the loop;

2. A runtime monitoring mechanism that is based on visual world model backbone and comes with task-adaptive error prediction thresholds;

3. The showcase of high performance of such multi-task fleet learning system both in simulation and on real robots.

[1]FIREWORK: **F**leet-based **I**nteractive **R**obot L**E**arning with Visual **WOR**ld Models for Multi-tas**K** Setting.

2 Related Work

Multi-task Robot Learning. There has been significant progress in the development of multi-task robotics agents [2, 26, 27, 1, 28, 29, 30, 31], driven by advancement in modern advanced policy architectures like transformer [1, 32] and diffusion models [33, 34], collection of larger robotics datasets [35, 36, 37, 38] and advancing robotics benchmarks [32, 39]. Instead of performing on task-specific domains, these models are generally capable to perform a diverse range of tasks, where the tasks can be specified commonly from language [29, 1] or multimodal prompts [27, 26].

Robot Fleet Learning. There has been growing interest in robot fleet learning [40, 4, 5, 41, 42, 6] as to scale up real-world robot deployment and data collection [43, 44, 45]. Prior literature has studied the different problems in fleet learning. Hoque et al. [14] proposed the interactive fleet learning (IFL) framework and studies the *resource allocation* problem [40]. Works in *decentralized and federated learning* [5, 41, 42] studies policy merging and data selection from individual fleets. Other works studies the *system* aspect of managing a robot fleet [6]. In this work, we studies the learning aspect of learning from the multi-task deployment data aggregated from a fleet of robot performing diverse tasks [46, 43].

Interactive Imitation Learning. To ensure trustworthy deployment and help robot learn on the job, human-in-the-loop methods [10, 9, 7, 13, 15, 8, 11] has been proposed to use human monitoring and intervention in robot task execution processes. To further reduce human workload in monitoring, runtime monitoring and error detection techniques [12, 17, 47, 16, 48, 49] seek to detect failures and anomalies during robot task execution. Two main bodies of work are unsupervised out-of-distribution (OOD) detection [18, 50, 19, 51, 20, 21], and failure detection either with binary classification [52, 53, 54] or learning risk function estimate from trajectory data [12, 14]. Driven by the recent development of foundation models, there has been works that uses LLMs / VLMs as error detectors for LLM- / VLM-based policies [55, 56, 57], or for object detection [58]. These works have yet to show applicability to robot motor action manipulation policies in close-loop due to the need to run runtime monitoring with fast, consistent inference speed on real robots [1].

3 FIREWORK: Multi-Task Interactive Robot Fleet Learning

3.1 Background

3.1.1 Problem Formulation

We formulate multi-task interactive robot fleet learning as finite-horizon Markov Decision Process (MDP), where N robots operates in N independent MDPs. The robot i operate in the i-th MDP specified by $\mathcal{M}_i = (S, A, \mathcal{T}, H_i, \mu_i^0, R_i)$, where S and A are the state and action spaces of the robot, $\mathcal{T} : S \times A \to S$ is the transition dynamics, H_i is the horizon length, μ_i^0 is the initial state distribution, and $R_i : S \times A \to \mathbb{R}$ is the reward function. In a sparse-reward setting, R_i is replaced with a goal predicate $g_i : S \to \{0, 1\}$. The collection of $\{\mathcal{M}_i\}_{i=1}^N$ can be reformulated as one single MDP, with shared S, A, T. The data from all robots are aggregated to learn one unified multi-task policy $\pi(a \mid s, g_i)$ that maximizes the expected return: $\max_\pi J(\pi) = \mathbb{E}_{s_t, a_t \sim \pi, \mu_0} \left[\sum_{t=1}^H g_i(s_t) \right]$.

3.1.2 Multi-Task Interactive Fleet Deployment

We consider an interactive learning framework [7, 13, 17] for a fleet of robots [4], where learning and deployment happens iteratively with a human π_H in the loop. The robot bootstrap its initial policy π_1 by doing behavioral cloning from a set of human demonstrations, $\mathcal{D}^0$. During each round deployment round i, each robot perform task execution with π_i with runtime monitoring, supervised by the error predictors, E_i. During policy execution, E_i determines whether current state can potentially lead to error states. Upon detecting an potential error, it indicates to the human and ask humans to monitor the process. While monitoring, the human can chooses to actively intervene [59] and take over the control if necessary, letting $\pi_H(s, g)$ overriding $\pi(s, g)$. We add the trajectories

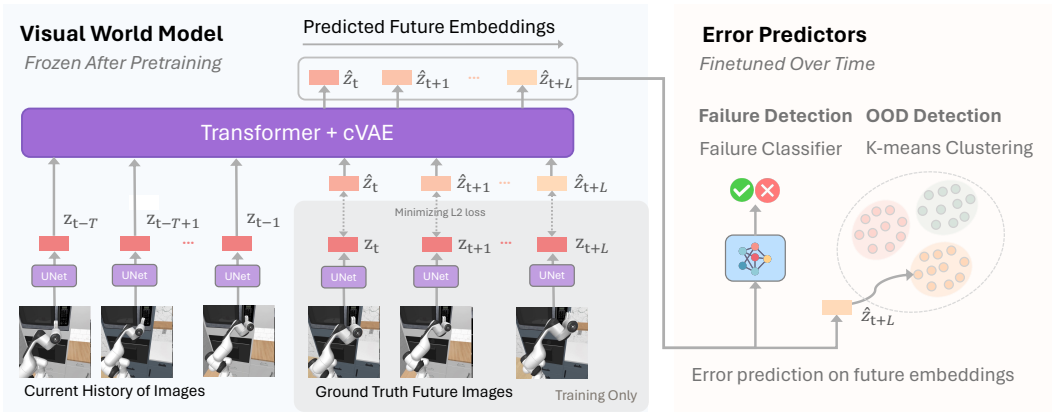

Figure 3: **Model Architecture.** The visual world model is trained with Unet encoder and decoder, as well a cVAE + Transformer based prediction model. The world model can be used to predict the future embeddings from the current state. The learned representations are used for error predictions, including failure and OOD detection.

$\tau = (s_t, a_t, r_t, c_t)$ to the data buffer of the current deployment round, $\mathcal{D}'$, where c_t indicates whether timestep t for trajectory τ is from human action. We update the data buffer $\mathcal{D}^{i+1} = \mathcal{D}^i \cup \mathcal{D}'$, which will be used to update the policy π_{i+1} and error predictors E_{i+1} for the next round.

3.2 Runtime Monitoring for Multi-Task Fleet Deployment

In this section, we discuss FIREWORK's runtime monitoring mechanism, which monitors different tasks simultaneously across different diverse scenes throughout the deployment. The key challenge is to efficiently monitor different tasks simultaneously.

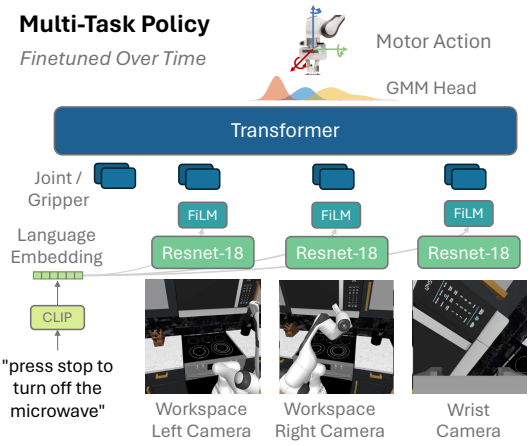

Figure 2: **Policy Architecture.** The multi-task policy is a transformer-based policy that takes in the image inputs (left workspace, right workspace, wrist), proprioceptive information, and the language embedding. It uses a GMM head and outputs robot motor action.

We envision such general runtime monitoring to meet two key design requirements: 1) generally-capable: the error predictor models should not be task-specific and should be efficiently trained and used for a variety of tasks; 2) task-adaptive: it should adapt to different tasks and their evolving progress during deployment. Additionally, as a runtime monitoring algorithm, it should be failure-preemptive, predicting the errors before they occur to ensure prompt detection and intervention.

Motivated by such design ideas, we build a multi-task runtime monitoring model that continuously adapts and improves for individual tasks during deployment. Our key idea is to train a visual world model backbone as a simulator for general task progress understanding, and use the backbone for downstream error predictions across different tasks. We present the model architecture in Figure 3.

3.2.1 Training the Backbone Visual World Model

Recent advancement of world models [22, 23, 24, 25] enables simulation into the future. Inspired by such progress, we train a visual world model on large diverse experience of robot trajectory frames.

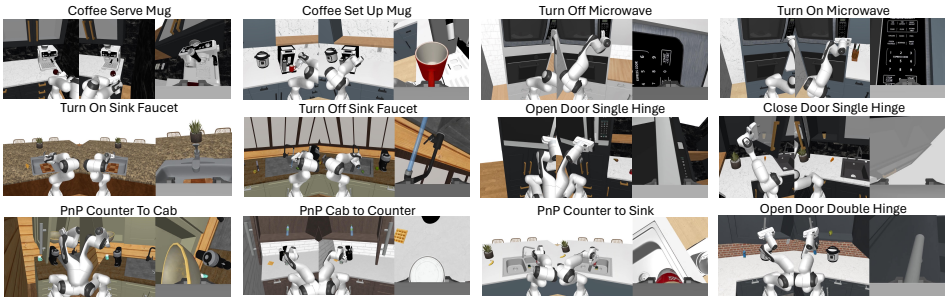

Figure 4: **Evaluation Tasks.** We show here all 12 simulation tasks from Robocasa; we will show the real-world Mutex tasks in the Appendix.

This enables future prediction of robot task progress, allowing us to see future outcome and preempt potential failures from happening.

FIREWORK trains an autoregressive visual world model $\mathcal{W}$ as the backbone for downstream error prediction. The world model $\mathcal{W} = (E_\gamma, D_\lambda, T_\psi)$ consists of an encoder E_γ, a decoder D_λ, and a conditional next state prediction model T_ψ. In training, E_γ encodes image observation x_t at timestep t into latent embedding z_t. The decoder D_λ reconstructs z_t to image $\hat{x_t}$, and minimizes the image reconstruction L2 loss. T_ψ inputs the history of T timesteps of embeddings $(z_{t-T}, z_{t-T+1}, \dots z_t)$, and outputs $\hat{z_{t+1}}$. It autoregressively predicts $(\hat{z_{t+1}}, \hat{z_{t+2}}, \dots \hat{z_{t+L}})$ for L steps into the future using the last T timesteps of embeddings and reconstructed embeddings, and minimizes the embedding reconstruction loss between $(\hat{z_{t+1}}, \hat{z_{t+2}}, \dots \hat{z_{t+L}})$ and $(z_{t+1}, z_{t+2}, \dots z_{t+L})$. E_γ and D_λ are implemented with UNet [60, 33], and T_ψ uses conditional Variational Autoencoder (cVAE) [61]. T_ψ is jointly trained with E_γ, D_λ on the same latent space.

We use a stochastic latent space rather than a deterministic one, since the stochastic latent space of cVAE supports multiple future sampling and facilitates better prediction [62, 17]. We use transformer architecture [63] for the encoder, decoder and prior network for the cVAE in T_ψ.

3.2.2 Building the Downstream Error Predictors

The visual world model gives information of how the world states change and how tasks progress over time. The learned representation from the world model can be useful for downstream error prediction tasks, where we predict into the future and see if the future state is an error. Since the individual tasks have different conditions, and can have different performance, we will train different error predictors on top of the frozen representation that are customized to the different tasks. We predict two kinds of errors: failure and our-of-distribution.

Failure Prediction. We train a failure classifier F_σ on the frozen latent space for each specific task. The failure labels are derived from the trajectories $\tau = (s_t, a_t, r_t, c_t)$ from the last round $\mathcal{D}'$ as detailed in Section 3.1.2, where c_t can indicate moments when human interventions occur. The segment of trajectory $(s_t, a_t, r_t, c_t)_{i-M}^{i-1}$ before the beginning of each human intervention s $c_i = \texttt{human}$ are used as failure labels [13, 17]. A three-class classification is used to distinguish between failure states, normal rollouts, and intervention states, so as to distinguish the nuanced difference between failures and interventions that often occur at similar states. The classifier is trained using a cross-entropy loss function $\mathcal{L}_\text{F} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$ with balanced sampling, where $y_i \in \{\texttt{normal}, \texttt{failure}, \texttt{human}\}$.

Out-of-Distribution (OOD) Prediction. We determine whether a state is OOD using k-means clustering. We use the frozen visual world model $\mathcal{W}$ to generate embeddings for a subset of trajectories sampling from the data buffer $\mathcal{D}^i$ at the deployment round i. We then reduce the dimensions of this latent space to l using Principal Component Analysis (PCA) and calculate c k-means centroids for each task. To perform OOD prediction for an embedding z, we first reduce its dimensions to l using PCA and find the nearest centroid of the last predicted latent space, and calculate their L2 distance d. The state is classified as OOD if d exceeds the task threshold, α_g. We determine this threshold

by calculating d_g^θ, the distance of the top θ_g percentile of the distances to the nearest centroids from the validation latent embeddings; we will explain how we determine θ in the following section. We observed that using k-means instead of the nearest neighbor approach [17] yields better space and time efficiency.

Adaptive Decision Boundary Auto-Adjustment. One challenge for multi-task runtime monitoring is that task performance varies across different tasks, and moreover task performance changes along deployment updates. We make our error predictors adapt to the varying performance of different tasks in the multi-task deployment. Our intuition is to assign looser decision boundaries to tasks with higher performance, which means the policy needs less supervision; and tighter boundaries to tasks with lower performance, allowing the error predictor to be more conservative and ask for more human monitoring to supervise the policy execution. We realize this idea for the failure predictor and OOD predictor respectively as follows. For the failure classifier, we fine-tune the classifier model on the deployment data from the most recent round, $\mathcal{D}'$, with the newest human intervention labels that reflects the most up-to-date risk perception of the robot on this task.

For the OOD detector, our intuition is that 1) we should enlarge the boundary for a state to be "out-of-distribution" since the robot has seen more states at deployment, and hence more states would be in-distribution; 2) the classification criteria changes as the robot policy improves over a task. When selecting embeddings to perform k-means clustering, we sample from D^i, which the past deployment data aggregated across previous rounds. As discussed in the previous section, to do OOD classification for an embedding z, we use a threshold α_g for its distance d to its nearest centroid, which depends on θ_g. Ideally, θ_g should depend on the policy performance on this task g. We use the human intervention ratio from the most recent round, p_H, as a proxy for the policy performance, and set θ_g to be a function of the human intervention ratio from the last round. We model this function as a exponential decay function, $\theta_g = a + be^{cp_H}$, and obtain the parameters by fitting the curve to a few test trajectories from different tasks to capture the relationship between the two values. We found this function to be robust across rounds or tasks, and we use the same function throughout the simulation and real world experiments. More discussion can be found in the Appendix.

3.2.3 FIREWORK in Operation

Here we discuss the design choices of training and running FIREWORK in operation.

Continual Model Improvement. The visual world model $\mathcal{W}$ is trained once and frozen afterwards, serving as a backbone for the general representation. Meanwhile, we want our system to continually improve over time. Therefore, the policy and the error predictors are being finetuned over deployment: the policy is continually finetuned on the aggregation of previous rounds of data with weighted sampling, the failure classifier is finetuned over the last round of intervention labels while the OOD classifier increases its latent space coverage every round.

Error Predictors in Operation. For failure and OOD predictions, the visual world model is used for inference to predict the latent embeddings L steps ahead of the current frame. We do the L-step prediction N times to sample N possible futures by drawing samples from the cVAE latent space. Each predicted future embedding is individually assessed by the error predictors. For the failure prediction, we average the failure label for each future embedding; for OOD prediction, we average the distances from each of the future states to their nearest clustering centroid, and compare the final average distance to the OOD threshold.

Training the Multi-Task Policy. We train a Transformer-based [63] multi-task policy shown in Figure 2, which inputs image observation, robot proprioceptive states, and task language goal, and output robot motor actions, following the design from robomimic [32] and robocasa [39].

4 Experiments

In our experiments, we seek to answer the following questions: How well does FIREWORK continue to improve its policy over deployment with effective runtime monitoring? 2) How does FIREWORK's

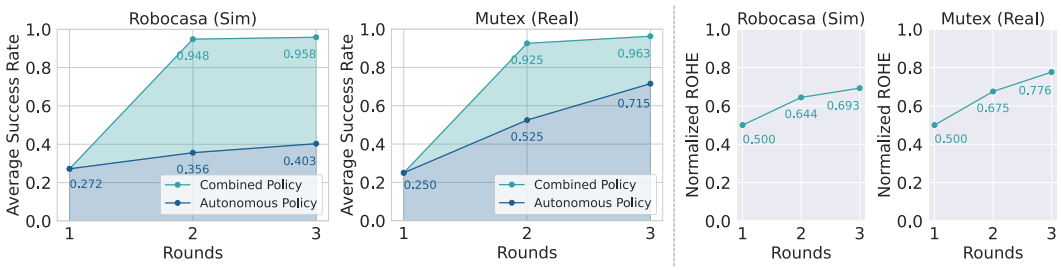

Figure 5: **System Performance Results.** The autonomous multi-task policy continues to improve over deployment. Left: Combined and Autonomous Policy Performance. Right: ROHE. Our runtime monitoring is able to ensure high combined policy performance as well as ROHE.

runtime monitoring prediction compare with baselines? 3) Are error prediction results predicted at the right timings qualitatively?

4.1 Evaluation Setup

To measure the effectiveness of the multi-task fleet learning, we want to see how the policy and runtime monitoring perform at the deployment and how the system performance evolves over time. Therefore, we evaluate the system performance in a human-in-the-loop setting of an iterative deployment loop, following prior works [64, 13, 17].

Evaluation Setting. Our evaluation process consists of 3 rounds of updates, with each round involving runtime-monitoring. The initial round requires full human supervision as there is no intervention data available to train error predictors at this time. For the following 2 rounds, human supervision is only needed when an error is predicted. For each round, the deployment data and intervention labels are collected to train the policy and error predictors for the next round.

Evaluation Metrics. To evaluate the policy performance, we utilize **Autonomous Performance** to measure how effectively the policy can achieve its goals without human intervention. For the effectiveness of runtime-monitoring, we use Combined Policy Performance and Return of Human Effort (ROHE) [12, 17]. **Combined Policy Performance** measures the overall system performance. **Return of Human Effort (ROHE)** evaluates how well the system utilizes human effort.

Evaluation Environments. We evaluate our system both in the simulator and real-world environment. For simulation, we use robocasa [39] as our simulator benchmark, a challenging benchmark due to its visually diverse tasks and environments with a wide range of objects, layouts, and scenes. The visual world model is trained on all task data generated from mimicgen [38] (5k trajectories $\times$ 20 task suites). The policy is trained on 12 task suites with human demonstration data (50 trajectories per task). The policy takes two workspace camera images (left and right view), and the wrist camera image. For real world, we use mutex [27] as our real world benchmark. The visual world model is trained on 50 tasks from the dataset, and the policy is trained on 10 tasks from 5 task suites. The policy takes in one workspace camera image and one wrist camera image.

	MoMaRT	PATO	ThriftyDAgger	FIREWORK
Combined Policy Performance	0.567	0.873	0.733	**0.958**
ROHE	0.434	0.522	0.518	**0.693**

Table 1: **Baseline Results Comparison.** FIREWORK surpassed the baselines both in combined policy performance and ROHE.

4.2 Evaluation Results

System Performance Over Time. We present our system-level performance results in Figure 5. Our first finding is that the autonomous policy consistently improves over the deployment rounds, with 13% increase in simulation and 45% in the real world. Second, FIREWORK's combined policy performance is consistently high, getting higher than 95% in both simulation and in real world. This shows that the runtime monitoring mechanism is effective at monitoring the deployment and

requesting human help when necessary. Third, FIREWORK also displays better ROHE performance over time. With finetuning from the deployment data, and a adaptive threshold, the error predictors At the same time, the improving policy also fails less, which makes the error predictors predict fewer errors. This shows the interplay of better policy trained from the deployment data and more confident error predictors.

Baseline Comparison. How does FIREWORK's multi-task runtime monitoring module with visual world model compare with prior methods? We compare FIREWORK with three prior works in runtime monitoring: **MoMart** [20], **PATO** [21] and **ThriftyDAgger** [12]. **MoMart** perform OOD detection using the VAE reconstruction loss of current input images. **PATO** performs OOD detection by combining the variance of ensemble policy and the variance of VAE reconstruction of future image goals. **ThrifyDAgger** combines OOD detection and failure detection, by measuring the the variance of ensemble policy and using a risky Q function to estimate the risky value of the current state. Since the baselines are originally designed for single task environments, we evaluate with 5 tasks in robocasa, each chosen from the 5 different task categories. For each baseline, we train a separate model for each of the tasks. We use the same multi-task policy as FIREWORK, and compare using the runtime monitoring metrics for one deployment round. We implement these baselines following their original designs. Speicifically, **MoMart** and **PATO** have a defined fixed threshold for their detection criteria. **ThriftyDagger**'s threshold is defined by the desired intervention ratio [12]. It calculate the risk value threshold based on the $(1-K)$-th percentile of all risk values in the data, where K is the desired human intervention ratio. For a fair comparison, we use the same human intervention ratio as FIREWORK in the same deployment round.

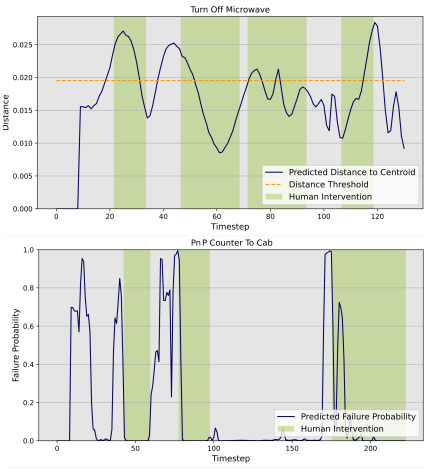

Figure 6: **Visualization of the error predictor timings compared with human intervention.** Green area indicates actual human intervention. Top: OOD Prediction. Bottom: Failure Prediction.

Table 1 shows the comparison of runtime monitoring metrics of FIREWORK with the baselines. FIREWORK surpasses the baselines both in combined policy performance and ROHE. One of our observation is that one fixed threshold value does not work for all the tasks. Due to the difference in task distribution and performance, the same threshold might be predicting all intervention for one task and zero intervention for the other, incurring high human workload for one and hurting task performance in another.

Analysis and Discussion. Can FIREWORK catch important errors and give meaningful intervention timings? We conduct qualitative analysis of where our error predictors predict error during robot execution, using active human monitoring. Specifically, we have a human operator fully supervise the robot execution of a policy and can intervene whenever an unsafe state is observed. We then apply the learned error predictors to the collected trajectories, and compare failure states identified by the error predictors with those intervened by the human operator. Figure 6 shows examples of the human intervention region and OOD / Failure errors predicted by FIREWORK, corresponding to the human risk assessment. We present more model analysis and discussion in the Appendix.

5 Conclusion

We presented FIREWORK a framework of multi-task interactive robot fleet learning. This consists of a multi-task robot policy and a runtime monitoring mechanism with visual world model backbone and task-adaptive error prediction thresholds. We showed the high performance of this multi-task fleet learning system both in simulation and on real robots.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2022.

[2] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent, 2022.

[3] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation, 2023.

[4] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg. Fleet-dagger: Interactive robot fleet learning with scalable human supervision. In *Conference on Robot Learning*, pages 368–380. PMLR, 2023.

[5] L. Wang, K. Zhang, A. Zhou, M. Simchowitz, and R. Tedrake. Robot fleet learning via policy merging, 2024.

[6] M. Müller, S. Brahmbhatt, A. Deka, Q. Leboutet, D. Hafner, and V. Koltun. Openbot-fleet: A system for collective learning with real robots, 2024.

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

[8] E. Chisari, T. Welschehold, J. Boedecker, W. Burgard, and A. Valada. Correct me if i am wrong: Interactive learning for robotic manipulation. In *RAL*, volume 7, pages 3695–3702, 2021.

[9] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *16th Robotics: Science and Systems, RSS 2020*. MIT Press Journals, 2020.

[10] 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. doi:10.1109/icra.2019.8793698. URL http://dx.doi.org/10.1109/ICRA.2019.8793698.

[11] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg. Lazydagger: Reducing context switching in interactive imitation learning. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 502–509. IEEE, 2021.

[12] 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.

[13] 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. In *Robotics: Science and Systems (RSS)*, 2023.

[14] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg. Fleet-dagger: Interactive robot fleet learning with scalable human supervision, 2022.

[15] Q. Li, Z. Peng, and B. Zhou. Efficient learning of safe driving policy via human-ai copilot optimization, 2022.

[16] 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)*, pages 1661–1667, 2019. doi:10.1109/IROS40897.2019.8968498.

[17] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu. Model-based runtime monitoring with interactive imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[18] 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*, abs/2110.14051, 2021. URL https://arxiv.org/abs/2110.14051.

[19] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. *Robotics: Science and Systems*, 07 2017.

[20] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1367–1378, 2022.

[21] S. Dass, K. Pertsch, H. Zhang, Y. Lee, J. J. Lim, and S. Nikolaidis. Pato: Policy assisted teleoperation for scalable robot data collection, 2022.

[22] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh. Video generation models as world simulators. 2024.

[23] J. Bruce, M. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, Y. Aytar, S. Bechtle, F. Behbahani, S. Chan, N. Heess, L. Gonzalez, S. Osindero, S. Ozair, S. Reed, J. Zhang, K. Zolna, J. Clune, N. de Freitas, S. Singh, and T. Rocktäschel. Genie: Generative interactive environments, 2024.

[24] J. Xiang, G. Liu, Y. Gu, Q. Gao, Y. Ning, Y. Zha, Z. Feng, T. Tao, S. Hao, Y. Shi, Z. Liu, E. P. Xing, and Z. Hu. Pandora: Towards general world model with natural language actions and video states. 2024.

[25] A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado. Gaia-1: A generative world model for autonomous driving, 2023.

[26] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

[27] R. Shah, R. Martín-Martín, and Y. Zhu. Mutex: Learning unified policies from multimodal task specifications. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=PwqiqaaEzJ.

[28] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[29] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022.

[30] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, A. Laurens, C. Fantacci, V. Dalibard, M. Zambelli, M. Martins, R. Pevceviciute, M. Blokzijl, M. Denil, N. Batchelor, T. Lampe, E. Parisotto, K. Żołna, S. Reed, S. G. Colmenarejo, J. Scholz, A. Abdolmaleki, O. Groth, J.-B. Regli, O. Sushkov, T. Rothörl, J. E. Chen, Y. Aytar, D. Barker, J. Ortiz, M. Riedmiller, J. T. Springenberg, R. Hadsell, F. Nori, and N. Heess. Robocat: A self-improving generalist agent for robotic manipulation, 2023.

[31] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy, 2024.

[32] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021.

[33] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.

[34] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024.

[35] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.

[36] Open X-Embodiment Collaboration et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024.

[37] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, et al. Droid: A large-scale in-the-wild robot manipulation dataset, 2024.

[38] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[39] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[40] G. Swamy, S. Reddy, S. Levine, and A. D. Dragan. Scaled autonomy: Enabling human operators to control robot fleets, 2020.

[41] O. Akcin, P.-h. Li, S. Agarwal, and S. P. Chinchali. Decentralized data collection for robotic fleet learning: A game-theoretic approach. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 978–988. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/akcin23a.html.

[42] O. Akcin, O. Unuvar, O. Ure, and S. P. Chinchali. Fleet active learning: A submodular maximization approach. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1378–1399. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/akcin23a.html.

[43] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, 2018.

[44] A. . Team, J. Aldaco, T. Armstrong, R. Baruch, J. Bingham, S. Chan, K. Draper, D. Dwibedi, C. Finn, P. Florence, S. Goodrich, W. Gramlich, T. Hage, A. Herzog, J. Hoech, T. Nguyen, I. Storz, B. Tabanpour, L. Takayama, J. Tompson, A. Wahid, T. Wahrburg, S. Xu, S. Yaroshenko, K. Zakka, and T. Z. Zhao. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation, 2024.

[45] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, I. Leal, S. Maddineni, K. Rao, D. Sadigh, P. Sanketi, P. Sermanet, Q. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

[46] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, 2016.

[47] K.-C. Hsu, H. Hu, and J. F. Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. 2023. URL https://api.semanticscholar.org/CorpusID:261697421.

[48] 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*, 2023.

[49] K. Nakamura, R. Tian, and A. V. Bajcsy. A general calibrated regret metric for detecting and mitigating human-robot interaction failures. *ArXiv*, abs/2403.04745, 2024. URL https://api.semanticscholar.org/CorpusID:268264177.

[50] R. Sinha, A. Sharma, S. Banerjee, T. Lew, R. Luo, S. M. Richards, Y. Sun, E. Schmerling, and M. Pavone. A system-level view on out-of-distribution data in robotics, 2022.

[51] 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)*, pages 5041–5048. IEEE, 2019.

[52] 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*, 228(8):1444–1458, 2014.

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

[54] A. Xie, F. Tajwar, A. Sharma, and C. Finn. When to ask for help: Proactive interventions in autonomous reinforcement learning, 2022.

[55] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners, 2023.

[56] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction, 2023.

[57] L. Guan, Y. Zhou, D. Liu, Y. Zha, H. B. Amor, and S. Kambhampati. "task success" is not enough: Investigating the use of video-language models as behavior critics for catching undesirable agent behaviors, 2024.

[58] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. Nesnas, and M. Pavone. Semantic anomaly detection with large language models, 2023.

[59] A. Hu, G. Corrado, N. Griffiths, Z. Murez, C. Gurau, H. Yeo, A. Kendall, R. Cipolla, and J. Shotton. Model-based imitation learning for urban driving, 2022.

[60] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[61] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

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

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[64] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *CoRL*, pages 879–893, 2018.