Fail2Progress: Learning from Real-World Robot Failures with Stein Variational Inference

Yixuan Huang[1], Novella Alvina[1], Mohanraj Devendran Shanthi[1], Tucker Hermans[1,2]
[1]University of Utah, [2]NVIDIA Research

Abstract: Skill effect models for long-horizon manipulation tasks are prone to failures in conditions not covered by training data distributions. Therefore, enabling robots to reason about and learn from failures is necessary. We investigate the problem of efficiently generating a dataset targeted to observed failures. After fine-tuning a skill effect model on this dataset, we evaluate the extent to which the model can recover from failures and minimize future failures. We propose Fail2Progress, an approach that leverages Stein variational inference to generate multiple simulation environments in parallel, enabling efficient data sample generation similar to observed failures. Our method is capable of handling several challenging mobile manipulation tasks, including transporting multiple objects, organizing a constrained shelf, and tabletop organization. Through large-scale simulation and real-world experiments, we demonstrate that our approach excels at learning from failures across different numbers of objects. Furthermore, we show that Fail2Progress outperforms several baselines. Qualitative results are available at sites.google.com/view/fail2progress.

Keywords: Learning from failures, Variational inference, Skill effect models

1 Introduction

Learned models of skill effects [1, 2, 3, 4, 5] show promising results in solving long-horizon manipulation tasks via skill sequencing. To train these models, researchers typically leverage simulation to efficiently generate large-scale, diverse data. However, robots using skill-based models in unstructured and uncertain real-world environments will inevitably struggle in out-of-distribution scenarios that are significantly different from the training datasets. In response, we want our robots to detect failures, recover from failures, and learn to minimize future failures so that they can continuously adapt once deployed.

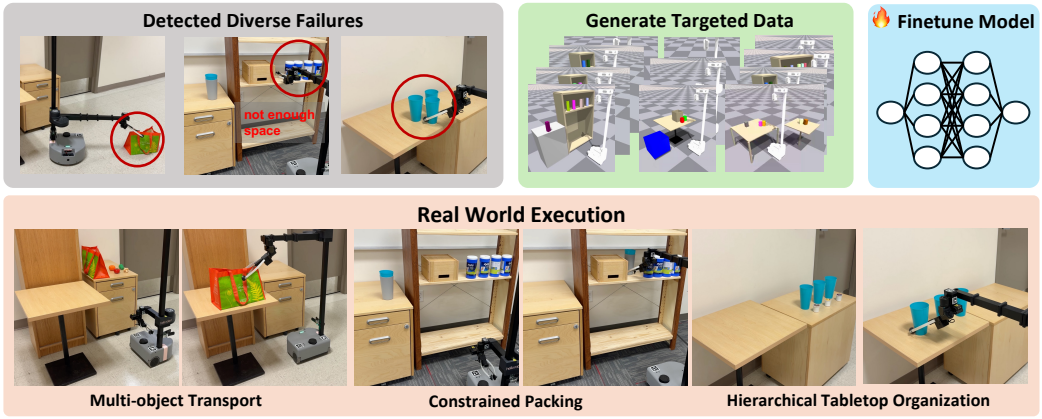

Figure 1: Overview of Failure Case Reasoning. Top: Based on failure cases due to incorrect symbolic predictions, our approach generates targeted, diverse simulation data in parallel to fine-tune the robot's model. **Bottom**: The fine-tuned model successfully performs diverse long-horizon manipulation tasks in challenging real-world scenarios.

Skill-effects models predict the change in world state when running a skill given an initial observation and continuous parameters associated with the skill. These effects can be full metric states, such as

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

the poses of objects, or symbolic states such as inter-object relations, logical states, or preconditions of other skills. Within this paradigm, we define a symbolic-level skill execution failure to occur when the world symbolic state after execution does not match the predicted (i.e. planned) symbolic effect state (i.e. incorrect symbolic predictions). When operating with symbolic states, the robot does not need to perfectly match any predicted metric state as long as the high-level sub-goal is reached.

Assuming the robot itself does not break and other agents do not disturb the environment, we can categorize failures into two types: (1) those arising from incorrect symbolic predictions (e.g., Fig. 1) and (2) those resulting from a Sim-to-Real (Sim2Real) gap during closed-loop skill execution. Specifically, if the system achieves the desired symbolic outcome, it indicates there is no skill-symbolic Sim2Real gap, even if the robot acts somewhat differently in the real world than in the simulator. If a failure does occur, it arises either because the trained model would incorrectly predict effects in an equivalent simulation scenario or because the closed-loop execution in simulation deviates significantly from its real-world execution. The Sim2Real gap could be caused by real-world perception noise, controller mismatch, or inaccurate physical modeling in the simulation, among other causes (details in Appx. A.7). In this paper, we investigate detecting failures, classifying failure types, and learning from failures due to incorrect symbolic predictions. Note that our approach is complementary to other works [6, 7, 8] addressing the Sim2Real gap.

While a robot could learn directly from real-world failure cases, a single failure instance is insufficient to effectively refine modern large parameter models [9, 10, 11]. The robot could instead try to explicitly generate more real-world failure scenarios [12, 13], but making the robot explore in the open environment poses risks to the robot and the surrounding environment. To address this, Real-to-Sim (Real2Sim) approaches [14, 15, 16] have gained popularity in robot manipulation, as they enable safe and efficient creation of simulation environments. Nevertheless, current methods emphasize high-fidelity simulations, which can be computationally expensive and require extensive fine-tuning [14, 15] or environment scanning [16]. Therefore, generating diverse data conditioned on failure cases efficiently and safely to improve skill effect models is an important and open question to address.

In this work, we advocate for generating low-fidelity simulation environments informed by real-world failures. We can generate such simulation datasets efficiently and safely to fine-tune skill effect models to minimize future failures in long-horizon tasks. Recent progress in physical simulation [17, 18] has shown success in accelerating simulation by running multiple environments in parallel on graphics processing units (GPUs). To leverage the power of parallel simulation, we propose to generate multiple simulation states in parallel. To this end, we formulate a variational inference problem to generate datasets targeted to observed failures for use in refining a skill effect model. To efficiently generate samples in parallel, we propose using Stein variational inference (SVI) [19] as our variational solver.

We introduce Fail2Progress, which employs SVI to generate a simulation dataset informed by failure cases to enhance the skill effect model. When the robot detects a real-world failure occurrence, it records the relevant state information (e.g., object relations), observation, and the executed robot skill associated with the failure. Given this, Fail2Progress generates a simulation dataset that approximates the joint distribution of states that match the observed failure and actions that maximize the robot's information gain [20] of its current skill effect model. The robot then fine-tunes its skill effect model on this dataset to improve its future handling of scenarios similar to the observed failure, as shown in Fig. 1.

In summary, our contributions are: (1): the formulation of a two-category failure classification problem between incorrect symbolic predictions and a Sim2Real gap. (2): We are the first to formulate failure case reasoning for long-horizon manipulation tasks as a variational inference problem, enabling the generation of diverse simulation data to improve the skill effect model. (3): We propose using Stein variational inference to approximate multi-modal posterior distributions over simulation states and robot skill parameters, thereby facilitating effective and efficient dataset generation. (4): We implement our proposed approach with three distinct skill effect model formulations [2, 3, 4], efficiently generating low-fidelity simulation environments. Through large-scale mobile manipulation experiments, we demonstrate that Fail2Progress outperforms several baselines.

2 Related Work

Failure case reasoning has gathered significant attention in the robotics community [21, 22, 23, 24, 25, 26, 27, 28]. However, these works focus on detecting failures and recovering from failures, but do not explicitly address the problem of learning to minimize future failures. Thomason and Kress-Gazit [29] propose to automatically improve a symbolic abstraction of a robot skill from observed failures. In contrast, we focus on improving pretrained skill effect models. Kumar et al. [12] enable the robot to explore real-world environments to improve its model. While effective, their approach is limited to closed environments and faces safety and efficiency challenges when generalizing to open, unstructured settings.

Real-to-Sim approaches show promise in efficiently generating large-scale simulation datasets for policy learning [16, 30, 31, 15, 32], and improving simulation physical parameters using real-world data [33, 34, 35, 8, 36]. Additionally, there is extensive literature on generating articulated objects from real-world 2D or 3D data [37, 15, 14, 38, 39, 40]. However, these approaches primarily focus on creating high-fidelity simulations, which can be computationally inefficient.

Stein variational inference (SVI) has found application in robotics due to its ability to approximate high-dimensional, multimodal posterior distributions [19, 41, 42, 43, 44, 45, 46]. Existing use cases focus on generating diverse and robust plans under uncertainty in various contexts [41, 45, 19, 42]. We instead use SVI to generate diverse simulated datasets to improve learned models.

3 Skill Effect Models

We build our failure reasoning approach on top of skill effect models [2, 3, 4] that can solve long-horizon, geometrically complex tasks directly from high-dimensional, partial-view point clouds. Our skill effect models require segmented point clouds and thus assume a perception pipeline with (1): the semantics of objects and (2): the segmentation of each object. In this paper, we use open-source models for segmentation [47] and detection [48]. We assume a given set of manipulation skill primitives such as push, pick and place, etc. $\mathcal{L} = \{\phi^1, ..., \phi^K\}$. Each primitive ϕ^k is parameterized with a continuous parameter $a^k \in \mathbb{R}^m$. A robot skill $\phi^k(a^k)$ is a skill primitive parameterized by a^k that can be executed on the robot. A set of relations $\mathcal{R} = \{\texttt{on}, \texttt{left}, \texttt{right}, ...\}$ is also given, the relations include inter-object relations or single-object relations. Inter-object relations include spatial relations like `left` and physical relations like `in-contact`. Single-object relations include whether an object is `manipulable` (e.g., a shelf is not manipulable) and whether a drawer is `open`. The plan skeleton $\phi_{1:H}$ is defined as a sequence of feasible primitives. When the plan skeleton is paired with valid continuous parameters, it enables the robot to achieve the goal relations from the initial state.

A skill effect model can be trained with a large-scale simulation dataset $\mathcal{D} = \{(s_t, O_t, \phi_t, a_t, s_{t+1}, O_{t+1})\}$, where O_t represents the observation at time t as segmented point clouds and s_t represents the simulation state information including geometric information like object pose and physical information like the object friction parameters and ϕ_t represents the skill to execute on the simulated robot with corresponding continuous parameters a_t. Given the initial simulation state s_t, we can get the effects of the skill $\phi_t(a_t)$. The ground-truth relations r_t are a function of the simulation state s_t. We train the model, Γ, with the dataset, $\mathcal{D}$. The trained model can predict the probability of achieving specific relations based on an initial point clouds observation, O_0, a robot skill sequence, and a training dataset as $\Gamma(r|O_0, \phi_{1:H}, a_{1:H}, \mathcal{D})$, where $r \in \mathcal{R}$. Given a goal, $\mathcal{G}$, defined as a conjunction of desired relations $g_1 \wedge ... \wedge g_M$, $g_i \in \mathcal{R}$. The planning objective is to find a skill sequence $\phi_{1:H}(a_{1:H})$ that maximizes the probability that the goal relations are achieved $p(\mathcal{G}|O_0, \phi_{1:H}, a_{1:H}, \mathcal{D})$, where the plan skeletons $\phi_{1:H}$ could be generated using any number of techniques such as: foundation models [2, 49], graph search [50], or other classical planners [51]. Given the skeleton, one can maximize the planning objective using standard numerical optimization techniques such as a shooting method [2, 5, 49] or a cross-entropy method [50, 52, 53]) to generate continuous parameters $a_{1:H}$. For more details (e.g., implementation details) about the skill effect model, please refer to Appx. A.11.

4 Detecting and Classifying Failures Autonomously

Consider a robot operating using a skill effect model Γ trained on some dataset, $\mathcal{D}$. A user tasks the robot to achieve a goal, $\mathcal{G}$. Given an initial observation, O_0, its skill effect model, and the goal, the robot plans a skill sequence, $\phi_{1:H}(a_{1:H})$. In addition to the skill sequence itself the skill effect model predicts a sequence of expected relations, $\mathcal{R}'_k$ (i.e. symbolic states) that the robot will observe after executing each skill in its sequence for $k=1,...,H$. The robot now begins executing its skills following the plan. After each skill execution in the sequence, the robot can observe the current scene, O_k, and detect the current symbolic state as $\hat{\mathcal{R}}_k$. When the observed relations, $\hat{\mathcal{R}}_k$, don't match the predicted relations, $\mathcal{R}'_k$, the robot detects that it has failed to achieve the current subtask.

In the case of a detected failure, the robot stores the associated failure event in order to learn from it. We define a failure event to include the relevant observations, relations (i.e. symbolic states), and skill associated with the failure: $F=(O^F=O_{k-1},\mathcal{R}^F=\hat{\mathcal{R}}_{k-1},\phi^F=\phi_k,\mathcal{R}^{F\prime}=\hat{\mathcal{R}}_k)$. Once the robot has detected a failure, it will classify the failure category. It first reconstructs the same simulation based on observation O^F and predicts the relational effects of the action $\phi_k(a_k)$ as $\mathcal{R}''_k$. If the simulation relational effects $\mathcal{R}''_k$ match the real-world relational effects $\mathcal{R}'_k$, then the robot will classify this failure as stemming from incorrect symbolic predictions. Otherwise, this failure is classified as a Sim2Real gap. We now consider the problem of learning from the failure instance to improve the skill effect model Γ.

5 Generating Targeted Datasets to Learn from Failure

Since modern, large neural networks typically cannot learn from a single failure instance [9, 10, 11], we pose the problem of learning from failures as a problem of efficiently collecting a new dataset $\mathcal{D}^+$. Determining which data points from an infinite possible set to generate and label can naturally be defined as an active learning problem [54, 55, 56, 20]. From this perspective, we can quantify the effectiveness of the generated additional training dataset $\mathcal{D}^+$ using the expected information gain criteria [20]. However, we want to target our new dataset to be similar to the scenario in which the robot failed, something which information gain alone does not address.

We thus define our problem as finding a dataset $\mathcal{D}^+$ that yields high expected information gain in terms of improvement in the predictions from Γ associated with the detected failure. At the same time, we ensure that the samples in $\mathcal{D}^+$ have a high probability of the same relations observed by the robot prior to executing the failed skill. Let us first define the form of a single sample $d_i^+ \in \mathcal{D}^+$, as $d_i^+=(s_i^+,O_i^+,\phi^F,a_i^+,s_i^{++},O_i^{++})$. Note that the skill, ϕ^F, is fixed for all samples to be the skill that failed to achieve the subtask. Next let us define the results of evaluating the sample action a_i^+ in the simulator f as $s_i^{++}=f(s_i^+,\phi^F,a_i^+)$, which when rendered defines the post skill observation $O_i^{++}=\Psi(s_i^{++})$. We also render pre-action states to observations as $O_i^+=\Psi(s_i^+)$. Thus, samples in $\mathcal{D}^+$ have only two free variables for us to search over: the initial simulator state, s_i^+; and the action to execute a_i^+. We will use S^+ to denote the set of state samples, $\{s^+\}$ in $\mathcal{D}^+$, and S to denote the random variable associated with the state. We will use a similar notation for actions and observations.

We can formalize our dataset generation problem as the following constrained optimization problem, noting that maximizing the expected information gain is equivalent to maximizing the KL-divergence between the predictive distributions of the updated model Γ^+ fine-tuned on $\mathcal{D}^+$, and the original model Γ

$$\underset{\mathcal{D}^+}{\text{argmax}} \quad D_{KL}\left(\prod_{r\in\mathcal{R}^{F\prime}}\Gamma^+(r|O,\phi^F,A,\mathcal{D}\cup\mathcal{D}^+)\,\middle\|\,\prod_{r\in\mathcal{R}^{F\prime}}\Gamma(r|O,\phi^F,A,\mathcal{D})\right) \tag{1a}$$

$$\text{subject to} \quad S^+\sim P(\mathcal{R}^F,O^F|S) \tag{1b}$$

Thus, we must find a set of simulator states, S^+, and actions A^+ which maximize the active learning objective, while also ensuring the sample states would generate the same relations and point cloud observations observed by the robot before the failure. This distribution in Eq. 1b factorizes as

$$P(\mathcal{R}^F,O^F|S)=\prod_{r\in\mathcal{R}^F}\Gamma(r|O=\Psi(S))P(O^F|S)P(S) \tag{2}$$

where the first term, $\Gamma(r|O = \Psi(S))$, encodes the objective of finding states in the simulator that achieve the same relations when rendered and evaluated by the skill effect model. The second term, $P(O^F|S)$, ensures that we generate point clouds that match the failure observation. The final term, $P(S)$, encodes a prior over valid states in the simulator.

This formulation presents several computational challenges. (1) The objective in Eq. 1a is intractable because there exists an infinite number of possible datasets, $\mathcal{D}^+$. (2) Evaluating Eq. 1 requires running the simulator to generate all samples in the putative $\mathcal{D}^+$ and retraining the skill effect model Γ for each possible dataset. (3) Finding simulator states s_i^+ that render to point clouds matching the failure observation, amounts to an inverse problem over object geometries and poses. (4) Ensuring that the states obey the constraint while maximizing the objective defines a high-dimensional, non-convex problem.

5.1 Approximate Constrained Expected Information Gain

We propose two specific approximations to the problem defined in Eq. 1 in order to make the problem tractable. We can summarize these approximations in the following problem:

$$\underset{S^+,A^+}{\operatorname{argmax}} \prod_{r\in\mathcal{R}^{F\prime}} H\big(\Gamma(r\,|\,\xi(S)O^F,\phi^F,A,\mathcal{D})\big) \tag{3a}$$

$$\text{subject to} \quad S^+ \sim \Gamma(r^F\,|\,O=\xi(S)O^F)P(S) \tag{3b}$$

Here we have replaced the expected information gain objective in Eq. 1a with the entropy defined over the epistemic uncertainty [57, 58] of the currently trained model, $\Gamma(\cdot, D)$, where $H(P(Y\,|\,X)) = -\sum_{y\in Y} P(Y=y\,|\,X)\ln P(Y=y\,|\,X)$. As a common approximation widely used in active learning [54, 55], this allows us to avoid running the simulator and fine-tuning at each iteration of dataset optimization, thereby simplifying the objective. The distribution defined in Eq. 2 implies that one needs to search over object poses and geometries that match the appearance of the partial-view point clouds observed during the failure event. This defines an infinitely large space of possible object shapes, which we wish to avoid searching over. Instead, we simplify the constraint to transpose the poses of the individual object point clouds in O^F, while ensuring that the point clouds still achieve the same relations when evaluated by the detector, i.e. Γ evaluated without any actions. We denote by $\xi(s)O^F$ the segment-wise transformation of the point cloud to the poses defined by the state vector s. Note, this allows us to search over object poses without using the full physics simulator or renderer. We use the simulator to generate $\mathcal{D}^+$ after finding S^+ and A^+. We describe how we instantiate this transformed point cloud to a full object for the simulator in Sec 5.3.

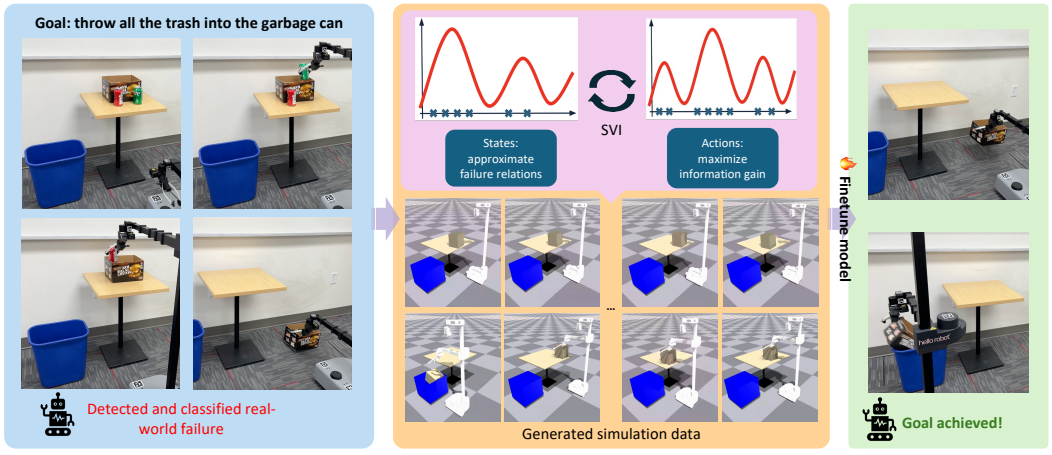

Figure 2: Overview of Fail2Progress. Our approach first detects the real-world failure and classifies it as incorrect symbolic predictions. Based on the real-world failure, Fail2Progress generates particles representing simulation states and actions to approximate posterior distributions. The state posterior distribution captures failure relations (e.g., objects inside the box), while the action posterior distribution maximizes the information gain of the skill effect model. Using these particles, a diverse simulation dataset is created to fine-tune the skill effect model. After fine-tuning, the model successfully recovers from the failure and completes the real-world task of cleaning.

5.2 Generating Datasets via Stein Variational Inference

We approximately solve the constrained optimization in Eq. 3 in two stages. First, we find a set of state samples S^+ that approximates the posterior distribution defined by Eq. 3b. We formulate finding this set as a variational inference problem. Then keeping S^+ fixed, we solve for the continuous action parameters A^+ that maximize Eq. 3a using generalized Bayesian inference [59, 19]. To solve both inference problems, we leverage Stein variational gradient descent which we now review.

Stein Variational Gradient Descent: In variational inference one defines a tractable distribution $q(X)$ to approximate the target distribution $P(X)$ [60]. One then optimizes over the parameters defining the variational distribution q in order to minimize the KL-divergence between the variational distribution and the target distribution $P(X)$ as $\underset{q(X)}{\operatorname{argmin}} D_{KL}\Big(q(X) \parallel P(X)\Big)$. Stein variational inference represents the posterior as a set of particles $q = \{x_i\}_{i=1}^M$. Stein variational gradient descent [61] (SVGD) leverages gradient-based optimization to guide the particles in a direction that minimizes the KL divergence. SVGD performs efficient approximate inference through parallel gradient-based optimization and can contend with high-dimensional and multi-modal posterior distributions. An SVGD particle x_i is updated at iteration k as $x_i^{+(k)} \leftarrow x_i^{+(k-1)} + \eta\Phi(x_i^{+(k-1)})$, where Φ is the Stein variational gradient computed using the Stein operator and a kernel $k(x_j^+, x_i^+)$. In this paper, we use a radial basis function (i.e. squared-exponential) kernel and set its kernel bandwidth using the median heuristic [19, 62].

In the case of generalized Bayesian inference (GBI), we modify the variational inference objective to account for defining a loss function over our variables instead of a traditional likelihood [59, 19]. Given a loss function $\mathcal{L}(Y, X)$ for an arbitrary random variable X and some observations Y, GBI defines the following approximate posterior $P_{\mathcal{L}}(X \mid Y) \propto P(x)\exp(-\beta\mathcal{L}(Y, X))$. We can then use this approximate posterior within a Stein variational inference framework by solving the following problem $\underset{q \in \mathcal{Q}}{\operatorname{argmin}} \beta\, \mathbb{E}_{X\sim q}[\mathcal{L}(X, Y)] + D_{KL}\Big(q(X) \,\Big\|\, P(X)\Big)$.

Generating State Samples: We want to find samples $q(S) = \{s_i^+\}_{i=1}^M$ that approximate the posterior distribution $P(r^F \mid O^+ = \xi(S)O^F)P(S)$, where $P(S)$ is a uniform prior over all feasible states. The posterior distribution ensures that the transformed point clouds match the relations in the failure case r^F. This defines the following variational inference problem: $\underset{q(S)}{\operatorname{argmin}} D_{KL}\Big(q(S) \,\Big\|\, \Gamma(r^F \mid O^+ = \xi(S)O^F)P(S)\Big)$ and it's solved by using SVI.

Generating Action Samples: Given our state samples generated using Stein variational inference to approximate the distribution in Eq. 3b, we can now turn our attention to solving for the action set A^+. To formulate this problem we make use of the generalized Bayesian inference framework outlined above. Here we define the loss function, $\mathcal{L}$, to be the entropy loss defined in Eq. 3a and let $\beta = 1$. The variational distribution takes the form $q(A) = \{(s_i^+, a_i^+)\}_{i=1}^M$, where we keep the values of s_i^+ fixed and search only over actions. This defines the following variational inference problem: $\underset{q(A)}{\operatorname{argmin}} \mathbb{E}_{s^+, a^+ \in q(A)}\left[\prod_{r\in\mathcal{R}^F} -H(\Gamma(r \mid \xi(s^+)O^F, \phi^F, a^+, D))\right] + D_{KL}\Big(A^+ \parallel P(A)\Big)$, where $P(A)$ is a uniform prior over actions. We solve the variational inference problem for generating state and action samples using SVI. We provide the details in Appx. A.13.

5.3 Real-to-Sim Object Generation

After running the two Stein inference procedures, we obtain optimized particles denoted as $\{s_i^+, a_i^+\}_{i=1}^M$. We now must generate the full dataset in simulation. First, we generate a simulation scene based on s_i^+ and then execute the corresponding robot skill, a_i^+. Since a failure observation, O^F, contains semantic segments for each object, we fit the corresponding object shapes (e.g., cuboids, open boxes, and drawers) based on these semantics. The bounding box of each segment determines the size of each object and combined with each object's pose, we construct the simulation scene.

We use pre-defined physical parameters (e.g., friction and center of mass) for each object class. After creating the simulation scene using s_i^+ and executing robot skill, (ϕ^F, a_i^+), we obtain our fine-tuning dataset, $\mathcal{D}^+$, to refine the skill effect model Γ. We note that our bounding-box-based real-to-sim scene generation is chosen primarily for its efficiency, allowing us to compare different dataset generation approaches. However, our primary contribution, Fail2Progress, is complementary to other real-to-sim methods [14, 16, 15]. If simulated objects generated by a Real2Sim pipeline do not capture important features of real objects, they may cause (1): incorrect classifications of failure types (incorrect symbolic predictions vs Sim2Real gap), and (2): poor fine-tuning data generation to improve the model. We examine the quality of our Real2Sim approach in the context of Fail2Progress in Sec. 6.

6 Experiments & Results

In this section, we present both simulation and real-world experiments to address several key questions: (Q1): To what extent does learning from failures improve long-horizon manipulation tasks? (Q2): Is adding more data sufficient to resolve all failure cases? (Q3): Can replanning alone effectively handle all failure scenarios? (Q4): How does Fail2Progress perform under noisy input? (Q5): Does SVI improve the performance of Fail2Progress? (Q6): Can Fail2Progress generalize to novel scenarios absent from both the pre-training and fine-tuning datasets? (Q7): Is Real2Sim accurate enough for our tasks?

We first use IsaacGym [17] to generate a pre-training dataset containing 40,000 skill executions, which is larger than the datasets used in [2, 3, 4]. After pre-training a skill effect model with this dataset, it is evaluated across diverse scenarios until a failure is detected. Based on the observed failure, Fail2Progress utilizes SVI to generate particles representing simulation states and corresponding robot actions. Once the simulation scenes are created and the robot skills are executed, Fail2Progress generates a targeted simulation dataset to fine-tune the skill effect model. Specifically, this targeted dataset consists of 20 diverse environments and their corresponding robot actions. We chose 20 data points based on an ablation study presented in Appx. A.5. Figure 2 visualizes an example simulation dataset generated by our approach.

Baseline Comparisons: In this paper, we compare our proposed approach, Fail2Progress, against six baselines on three state-of-the-art skill effect models [2, 3, 4]: **Original [2, 3, 4]:** These are the state-of-the-art skill effect models without fine-tuning. It serves as a baseline to demonstrate the performance of pure skill effect models without learning from out-of-distribution failures. **Small:** This baseline was trained using the pre-training dataset with a small set (40000+20 data points). **Large:** This baseline used the pre-training dataset alongside a much larger dataset (40000+2000 data points). **Gradient [56]:** This approach uses stochastic gradient descent to update individual particles, employing the same objective as Fail2Progress, while generating each sample sequentially and independently. This baseline serves to demonstrate the effectiveness of Fail2Progress. **Sampling [55]:** This sampling-based method iteratively generates each simulation state with the corresponding robot action until they satisfy the objective [63]. We use this baseline to highlight the challenges faced by sampling-based approaches in high-dimensional spaces. **Replanning [64]:** This approach relies exclusively on replanning to recover from failure cases. We include it to assess whether a pure replanning strategy is sufficient for failure recovery. We evaluate our approach against baselines in the tasks shown in Fig. 1. We provide detailed explanations of these tasks in Appx. A.2.

Simulation Success Rate Evaluation: We first compare Fail2Progress against all baselines on the **Hierarchical Tabletop Organization** task. We evaluate execution success rates across varying numbers of objects, as shown in Table 1. Each approach is tested with 300 trials. The comparison between Fail2Progress and Original demonstrates that Fail2Progress significantly outperforms Original, highlighting the importance of learning from out-of-distribution failures (Q1). The comparison shows that Fail2Progress outperforms Small and Large by a large margin, demonstrating that merely adding more data is insufficient to resolve all failure cases, highlighting the importance of our approach in selecting quality, targeted

	Original	Small	Large	Replanning	Sampling	Gradient	Fail2Progress
Points2Plans [2]	11%	13%	16%	24%	53%	45%	**86%**
Stow-GNN [3]	10%	11%	15%	21%	51%	44%	**80%**
Binary-Pred [4]	8%	9%	12%	19%	41%	38%	**72%**

Table 1: Simulation experiments for the **Hierarchical Tabletop Organization**. Fail2Progress outperforms all baselines by a large margin. Further details are reported in Appx. A.6.

data (Q2). While Replanning performs slightly better than Original, it falls far short of Fail2Progress, demonstrating that replanning alone is insufficient for recovering from failures, particularly in scenarios with large prediction errors in the dynamics model (Q3). Furthermore, the success rate comparisons among Fail2Progress, Gradient, and Sampling reveal that Fail2Progress consistently outperforms both baselines (Q5). This superior performance is attributed to SVI's capability to effectively approximate high-dimensional, multi-modal posterior distributions. Furthermore, the average relation detection F1 score of our approach is 0.92, ensuring reliable failure detection.

Generalization Evaluation: We assess the generalization capability of Fail2Progress compared to Gradient and Sampling in the **Multi-object Transport** task with the Points2Plans architecture. We evaluate generalization to an unseen number of objects and unseen viewpoints shown in Table 2. While all approaches experience some performance degradation, Fail2Progress maintains strong performance (Q6).

Generalization Scenarios	3objs ↑	5objs ↑	7objs ↑	view1 ↑	view2 ↑
Fail2Progress	**87%**	**81%**	**71%**	**83%**	**85%**
Gradient	51%	40%	18%	42%	44%
Sampling	62%	45%	23%	51%	47%

Table 2: We show the generalization capability of Fail2Progress in simulation across varying numbers of objects and different viewpoints. The 5objs, 7objs, view1, and view2 scenarios are unseen during training.

Real-world Quantitative Evaluation: We compare our approach, Fail2Progress, with the Gradient and Sampling baselines in real-world scenarios with the Points2Plans architecture. As shown in Fig. 3a, Fail2Progress consistently outperforms both baselines (Q5). For the real-world experiments, we performed 10 trials per approach for each object count. We also examine how the Sim2Real gap affects the performance of Fail2Progress, as shown in Fig. 3b. Without artificially added noise in the real-world point clouds, Fail2Progress successfully detects failures, classifies the failure reason as incorrect symbolic predictions, and achieves a high success rate shown in the left bar of Fig. 3b, which demonstrates that Sim2Real gap is small and Real2Sim is accurate enough (Q7). However, the Sim2Real gap becomes a significant issue when the real-world point clouds are noisy. To analyze this effect, we add random Gaussian noise to the point clouds. We quantify the Sim2Real gap as the difference in predicted relations between the real-world and reconstructed simulation scenarios, using the same robot skill. With noisier point clouds, the Sim2Real gap increases, leading to a degradation in the performance of Fail2Progress (Q4). We provide efficiency experiments (Appx. A.3), further qualitative analysis (Appx. A.1) and a summary of key findings (Appx. A.4) in the appendix.

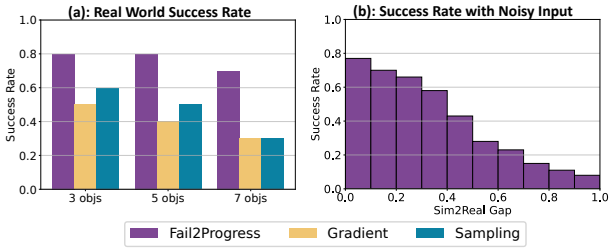

Figure 3: Fail2Progress consistently achieves higher success rates than all baselines on the **Hierarchical Tabletop Organization** task (a). Without artificially added noise in the point clouds, Fail2Progress performs well. However, its performance degrades under a large Sim2Real gap caused by noisy point clouds (b).

7 Conclusion

This work addresses the problem of learning from failures in long-horizon manipulation tasks using learned skill effect models. We propose generating additional, targeted simulation datasets based on observed failures to fine-tune the pre-trained skill effect model. We formalize the task as a probabilistic inference problem that maximizes the information gain of the datasets while ensuring the datasets remain close to the observed failure. To solve it, we introduce Fail2Progress, an approach that leverages SVI to approximate multi-modal posterior distributions. Through experiments, we demonstrate that Fail2Progress can generate failure-driven simulation datasets to improve the skill effect model more effectively and efficiently compared to six baselines. Furthermore, we deploy Fail2Progress on a mobile manipulator, showcasing its ability to perform diverse real-world tasks, such as packing groceries, packing a constrained shelf, and organizing a table.

8 Limitations

Our approach has several limitations. First, although Fail2Progress significantly improves performance, it still falls short of perfect reliability, achieving around an 80% success rate in the real world shown in Fig. 3a. This is because, even after fine-tuning, some scenarios remain out-of-distribution, leading to incorrect symbolic predictions. Indeed, one can think of the results presented in this paper as "1-shot" Fail2Progress and that further refinement on the observed failures would lead to higher future success rates. To continuously improve the performance as a lifelong learning system, the framework needs to be deployed in a real environment over several days, where we allow Fail2Progress to update as needed when failures are detected and classified as being caused by incorrect symbol predictions. Safely deploying Fail2Progress in such open environments remains an open research question. Furthermore, our framework needs to be evaluated under more diverse conditions, including more complex and dexterous manipulation tasks involving varied objects, such as deformable objects and liquids.

Second, we do not investigate correcting for failures caused by the Sim2Real gap in this work. The Sim2Real gap could potentially be mitigated by methods that explicitly address this challenge [6, 7, 8]. Showing how to integrate Sim2Real improvements alongside symbolic prediction failures is an important next step.

Third, we rely on Real2Sim to classify failures and generate high-quality fine-tuning datasets. Though our experiments show that our Real2Sim solution is effective in classifying failures and improving model performance, our Real2Sim itself is not perfect, especially when modeling complex object geometries and deformable objects.

Fourth, our failure classification scheme, which includes two categories, does not explicitly reason about the environmental disturbances caused by other agents (human users or other robots). It additionally does not account for hardware breaking or changing over time (e.g., cable or belt stretch in a robot arm drivetrain), which might occur over long deployment times. Hypothesizing these scenarios as failure causes is also an interesting future direction.

Fifth, we consider only object poses as the simulation state. Incorporating additional simulation states, such as object friction and center of mass [33], into our framework would be a possible next step.

Sixth, we assume a fixed set of relations. While our large-scale experiments show that these relations are sufficient, there are always relations outside the predefined set. Discovering new relations [65, 66] during robot exploration could enhance the open-world planning capability of our framework.

Finally, although we demonstrate mobile manipulation in diverse environments, extending the system to building-wide open spaces [67] remains an open research question. To achieve this, our method could integrate with scene graph construction and online updating [68, 69, 70, 71].

Acknowledgments

We would like to thank Jeannette Bohg and Christopher Agia for fruitful discussions. This work was partially supported by NSF Awards #2149585, #1846341, and #2321852, by DARPA under grant N66001-19-2-4035, and by a Sloan Research Fellowship.

References

[1] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer. Search-Based Task Planning with Learned Skill Effect Models for Lifelong Robotic Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. URL https://arxiv.org/abs/2109.08771.

[2] Y. Huang, C. Agia, J. Wu, T. Hermans, and J. Bohg. Points2plans: From point clouds to long-horizon plans with composable relational dynamics. *arXiv preprint arXiv:2408.14769*, 2024.

[3] H. Chen, Y. Niu, K. Hong, S. Liu, Y. Wang, Y. Li, and K. R. Driggs-Campbell. Predicting object interactions with behavior primitives: An application in stowing tasks. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=VH6WIPF4Sj.

[4] C. Paxton, C. Xie, T. Hermans, and D. Fox. Predicting Stable Configurations for Semantic Placement of Novel Objects. In *Conference on Robot Learning (CoRL)*, 11 2021. URL https://arxiv.org/abs/2108.12062.

[5] C. Agia, T. Migimatsu, J. Wu, and J. Bohg. STAP: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.

[6] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018.

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[8] F. Ramos, R. C. Possas, and D. Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[9] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui,

Z. Zhang, Z. Fu, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[10] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[11] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014.

[12] N. Kumar, T. Silver, W. McClinton, L. Zhao, S. Proulx, T. Lozano-Pérez, L. P. Kaelbling, and J. Barry. Practice makes perfect: Planning to learn skill parameter policies. In *Robotics: Science and Systems (RSS)*, 2024.

[13] L. Smith, Y. Cao, and S. Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10829–10836. IEEE, 2024.

[14] Z. Mandi, Y. Weng, D. Bauer, and S. Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024.

[15] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[16] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*, 2024.

[17] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. In *Advances in Neural Information Processing Systems*, 2021. URL https://sites.google.com/view/isaacgym-nvidia.

[18] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[19] J. Pavlasek, S. R. Lewis, B. Sundaralingam, F. Ramos, and T. Hermans. Ready, set, plan! planning to goal sets using generalized bayesian inference. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=5JMGq83yf1N.

[20] F. B. Smith, A. Kirsch, S. Farquhar, Y. Gal, A. Foster, and T. Rainforth. Prediction-oriented bayesian active learning. In *International Conference on Artificial Intelligence and Statistics*, pages 7331–7348. PMLR, 2023.

[21] A. Farid, D. Snyder, A. Z. Ren, and A. Majumdar. Failure prediction with statistical guarantees for vision-based robot control. *arXiv preprint arXiv:2202.05894*, 2022.

[22] A. Inceoglu, E. E. Aksoy, A. C. Ak, and S. Sariel. Fino-net: A deep multimodal sensor fusion framework for manipulation failure detection. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6841–6847. IEEE, 2021.

[23] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=yqLFb0RnDW.

[24] A. Sharma, N. Azizan, and M. Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. In *Uncertainty in artificial intelligence*, pages 1958–1967. PMLR, 2021.

[25] P. Antonante, D. I. Spivak, and L. Carlone. Monitoring and diagnosability of perception systems. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 168–175. IEEE, 2021.

[26] S. Vats, M. Likhachev, and O. Kroemer. Efficient recovery learning using model predictive meta-reasoning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[27] K. Namasivayam, A. Tuli, V. Bindal, H. Singh, P. Singla, and R. Paul. Learning to recover from plan execution errors during robot manipulation: A neuro-symbolic approach. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12632–12639. IEEE, 2024.

[28] S. Vats, D. K. Jha, M. Likhachev, O. Kroemer, and D. Romeres. Recoverychaining: Learning local recovery policies for robust manipulation. *arXiv preprint arXiv:2410.13979*, 2024.

[29] W. Thomason and H. Kress-Gazit. Counterexample-guided repair for symbolic-geometric action abstractions. *IEEE Transactions on Robotics*, 39(5):4152–4165, 2023.

[30] M. Torne, A. Jain, J. Yuan, V. Macha, L. Ankile, A. Simeonov, P. Agrawal, and A. Gupta. Robot learning with super-linear scaling. *arXiv preprint arXiv:2412.01770*, 2024.

[31] L. Wang, R. Guo, Q. Vuong, Y. Qin, H. Su, and H. Christensen. A real2sim2real method for robust object grasping with neural surface reconstruction. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2023.

[32] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg. Planar robot casting with real2sim2real self-supervised learning. *arXiv preprint arXiv:2111.04814*, 2021.

[33] M. Memmel, A. Wagenmaker, C. Zhu, P. Yin, D. Fox, and A. Gupta. Asid: Active exploration for system identification in robotic manipulation. *arXiv preprint arXiv:2404.12308*, 2024.

[34] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[35] L. Ma, J. Meng, S. Liu, W. Chen, J. Xu, and R. Chen. Sim2real 2: Actively building explicit physics model for precise articulated object manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11698–11704. IEEE, 2023.

[36] R. Antonova, J. Yang, P. Sundaresan, D. Fox, F. Ramos, and J. Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3): 5819–5826, 2022.

[37] S. Qian, L. Jin, C. Rockwell, S. Chen, and D. F. Fouhey. Understanding 3d object articulation in internet videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1599–1609, 2022.

[38] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

[39] E. Heiden, Z. Liu, V. Vineet, E. Coumans, and G. S. Sukhatme. Inferring articulated rigid body dynamics from rgbd video. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8383–8390. IEEE, 2022.

[40] Y. Mao, Y. Zhang, H. Jiang, A. Chang, and M. Savva. Multiscan: Scalable rgbd scanning for 3d environments with articulated objects. *Advances in neural information processing systems*, 35: 9058–9071, 2022.

[41] L. Barcelos, A. Lambert, R. Oliveira, P. Borges, B. Boots, and F. Ramos. Dual Online Stein Variational Inference for Control and Dynamics. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.068.

[42] Y. Lee, A. Z. Li, P. Huang, E. Heiden, K. M. Jatavallabhula, F. Damken, K. Smith, D. Nowrouzezahrai, F. Ramos, and F. Shkurti. Stamp: Differentiable task and motion planning via stein variational gradient descent. *arXiv preprint arXiv:2310.01775*, 2023.

[43] T. Power and D. Berenson. Constrained stein variational trajectory optimization. *IEEE Transactions on Robotics*, 2024.

[44] A. Lambert and B. Boots. Entropy regularized motion planning via stein variational inference. *arXiv preprint arXiv:2107.05146*, 2021.

[45] A. Lambert, B. Hou, R. Scalise, S. S. Srinivasa, and B. Boots. Stein variational probabilistic roadmaps. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11094–11101. IEEE, 2022.

[46] K. Honda, N. Akai, K. Suzuki, M. Aoki, H. Hosogaya, H. Okuda, and T. Suzuki. Stein variational guided model predictive path integral control: Proposal and experiments with fast maneuvering vehicles. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7020–7026. IEEE, 2024.

[47] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[48] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[49] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.

[50] Y. Huang, N. C. Taylor, A. Conkey, W. Liu, and T. Hermans. Latent Space Planning for Multi-Object Manipulation with Environment-Aware Relational Classifiers. *IEEE Transactions on Robotics (T-RO)*, 2024. URL https://arxiv.org/pdf/2305.10857.pdf.

[51] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021. URL https://arxiv.org/abs/2010.01083.

[52] Y. Huang, A. Conkey, and T. Hermans. Planning for Multi-Object Manipulation with Graph Neural Network Relational Classifiers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2209.11943.

[53] Y. Huang, J. Yuan, C. Kim, P. Pradhan, B. Chen, L. Fuxin, and T. Hermans. Out of Sight, Still in Mind: Reasoning and Planning about Unobserved Objects with Video Tracking Enabled Memory Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[54] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6 (1):1–114, 6 2012.

[55] A. Conkey and T. Hermans. Active Learning of Probabilistic Movement Primitives. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 10 2019. URL https://arxiv.org/abs/1907.00277.

[56] Q. Lu, M. V. der Merwe, and T. Hermans. Multi-Fingered Active Grasp Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10 2020. URL https://arxiv.org/abs/2006.05264.

[57] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

[58] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[59] T. Matsubara, J. Knoblauch, F.-X. Briol, and C. J. Oates. Robust generalised bayesian inference for intractable likelihoods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(3):997–1022, 2022.

[60] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017. doi:10.1080/01621459.2017.1285773. URL https://doi.org/10.1080%2F01621459.2017.1285773.

[61] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[62] D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

[63] W. R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(2):337–348, 1992.

[64] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023.

[65] N. Shah, J. Nagpal, P. Verma, and S. Srivastava. From reals to logic and back: Inventing symbolic vocabularies, actions and models for planning from raw data. *arXiv preprint arXiv:2402.11871*, 2024.

[66] A. Ahmetoglu, B. Celik, E. Oztop, and E. Ugur. Discovering predictive relational object symbols with symbolic attentive layers. *IEEE Robotics and Automation Letters*, 2024.

[67] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martín-Martín. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation. *arXiv preprint arXiv:2410.06237*, 2024.

[68] P. Liu, Z. Guo, M. Warke, S. Chintala, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Dynamem: Online dynamic spatio-semantic memory for open world mobile manipulation. *arXiv preprint arXiv:2411.04999*, 2024.

[69] Y. Tang, M. Wang, Y. Deng, Z. Zheng, J. Deng, and Y. Yue. Openin: Open-vocabulary instance-oriented navigation in dynamic domestic environments. *arXiv preprint arXiv:2501.04279*, 2025.

[70] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=wMpOMO0Ss7a.

[71] C. Agia, K. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022.

[72] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020. URL https://arxiv.org/abs/1802.08705.

[73] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Sample-based methods for factored task and motion planning. In *Robotics: Science and Systems*, 2017. URL https://dspace.mit.edu/bitstream/handle/1721.1/137701/garrett-rss17.pdf?sequence=2&isAllowed=y.

[74] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez. Learning to guide task and motion planning using score-space representation. *The International Journal of Robotics Research*, 38 (7):793–812, 2019. URL https://arxiv.org/abs/1807.09962.

[75] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1940–1946. IEEE, 2022.

[76] D. Driess, J.-S. Ha, and M. Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Proceedings of Robotics: Science and Systems*, 2020. URL https://arxiv.org/abs/2006.05398.

[77] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

[78] S. Cheng and D. Xu. League: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 2023.

[79] W. Wu, Z. Qi, and L. Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019. URL https://arxiv.org/abs/1811.07246.

[80] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[81] D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Appendix

Overview

The appendix provides additional details, experiments, and results. Please refer to the supplemental video for real-world robot executions available at sites.google.com/view/fail2progress.

A.1 Qualitative Analysis

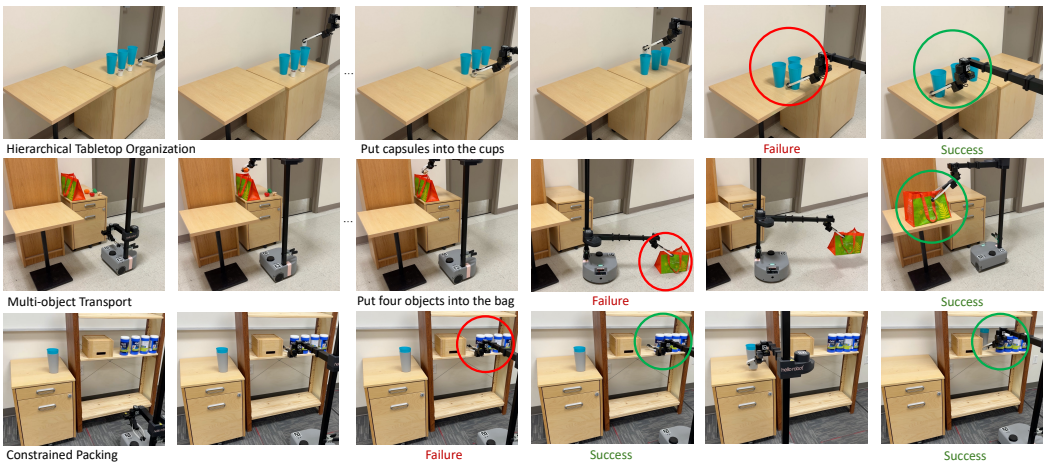

Figure 4: Rollouts of real-world evaluations and corresponding failure cases. A detailed explanation of this figure is provided in Sec. A.1.

Figure 5: Real-world generalization visualizations. We show how Fail2Progress generalizes to different numbers of objects (3-6), different object shapes, and different tables.

We present qualitative results in Fig. 4. **Hierarchical Tabletop Organization** task (First row): The robot is tasked with organizing the cups and capsules on another table while keeping them in a row. It first places several capsules into their corresponding cups. In the failure case, the robot fails to recognize the correlation between cups and capsules, resulting in the wrong organization. After learning from this failure, Fail2Progress successfully completes this task by understanding that the capsules will move with their corresponding cups. **Multi-object Transport** task (Second row): The robot is tasked with packing groceries and placing them on the table. It places all four groceries inside a bag. In the failure case, the robot places the bag on the ground instead of the table, failing the task. After fine-tuning the model with a targeted dataset, Fail2Progress moves the bag to the table. **Constrained Packing**

task (Third row): The robot is tasked with organizing a shelf by placing a stack of cups on a constrained shelf. In the failure case, the robot fails to make all the wipes in contact to clear enough space. After learning from the failure, Fail2Progress first pushes the wipes aside in contact to create sufficient space for the cups, then places them on the shelf.

Furthermore, we demonstrate how our approach generalizes to different numbers and shapes of objects, as well as different tables, in Fig. 5. Specifically, the model is fine-tuned only on failure cases with 3 objects but is able to generalize to scenarios involving 3-6 diverse objects on two tables.

A.2 Detailed Experimental Tasks

Multi-object Transport tasks the robot to transport multiple objects within a container using a single skill (e.g., carrying multiple fruits in a grocery bag). To succeed, the robot has to understand that all objects inside the container move together when the container is moved. **Hierarchical Tabletop Organization** tasks the robot to organize a table by arranging objects into a hierarchical structure (e.g., multiple objects in different cups). Success requires the robot to understand the relationships between these objects and how its skills impact future relations based on the hierarchical structure. **Constrained Packing** tasks the robot to organize objects in a constrained environment (e.g., a bookshelf). Success involves using a non-prehensile push skill to create space and then packing the remaining objects onto the shelf. In this paper, we present quantitative results for the **Multi-object Transport** and **Hierarchical Tabletop Organization** tasks, and qualitative results for the **Constrained Packing** task.

A.3 Efficiency Experiments

We compare Fail2Progress with the two best-performing baselines, Gradient and Sampling, to assess optimization efficiency using the best-performing architecture (Points2Plans). Error bars in the figure represent standard deviations across five different random seeds. As shown in Fig. 6, Fail2Progress is significantly more efficient than Sampling. This superior efficiency is attributed to the parallel computation capabilities of SVI on GPUs. Gradient achieves comparable efficiency to Fail2Progress, but it still performs significantly worse in terms of fine-tuned model performance shown in Fig. 3 and Table 1.

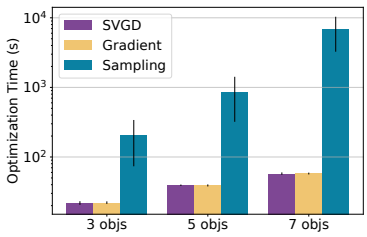

Figure 6: Efficiency experiments show that Fail2Progress is comparable to Gradient and more efficient than Sampling. The optimization time is presented on a logarithmic scale.

A.4 Key Findings

Importance of Learning from Failures: Learning from failures is essential because an initial training dataset for a skill effect model cannot capture all possible transitions in the real world. When the robot encounters novel scenarios outside the training data distributions, failures become inevitable. By learning from these failures, the robot can improve its performance more reliably and efficiently.

Limitations of Replanning: While Replanning can recover from certain failures, its effectiveness is inherently limited. It relies on the learned dynamics model, and any inaccuracies in the model can significantly degrade its performance.

Effectiveness of Fail2Progress with SVI: Through large scale comparisons, we demonstrate that Fail2Progress consistently outperforms the Sampling and Gradient baselines. This superior performance is attributed to SVI's ability to approximate high-dimensional posterior distributions using multiple particles. In contrast, Sampling does not leverage gradient information, making it highly inefficient. It also suffers from poor exploration with high-dimensional input. While Gradient uses gradient information, it suffers from mode collapse, leading to poor performance when the posterior distribution is multi-modal.

Efficiency of Fail2Progress with SVI: Generating additional simulation datasets from observed failures requires solving a high-dimensional and multi-modal posterior distribution inference problem. Fail2Progress, utilizing SVI, performs this task efficiently and achieves superior performance compared to Sampling. This efficiency is due to SVI's ability to approximate complex posterior distributions in parallel, leveraging GPU computational power.

Our experiments further reveal that as few as 20 targeted simulation data points are sufficient to fine-tune the skill effect model. This efficiency stems from the pre-trained model's ability to capture general representations that are transferable to related tasks. As a result, when the robot fails at a specific task, it can be efficiently fine-tuned on the new task to recover from failures.

Generalization of Fail2Progress: Through both simulation and real-world experiments, we demonstrate that our approach generalizes to varying numbers and shapes of objects, different environments (e.g., tables), and viewpoints beyond those in the fine-tuning dataset. This demonstrates that our framework does not overfit to specific scenarios but instead captures object interaction in a generalizable manner. By combining the ability to continuously learn from failures and generalize to unseen scenarios, we believe our framework can adapt to diverse and complex real-world household environments, assisting in daily tasks.

A.5 Ablation Study

To determine the best fine-tuning dataset size, measured by the number of particles, we conduct an ablation study presented in Fig. 7. The study is performed on the **Hierarchical Tabletop Organization** task with three objects, using Points2Plans as the underlying architecture. Through the ablation study, we find that increasing the number of particles generally improves the execution success rate of Fail2Progress but also increases the optimization time. Using 20 particles achieves the best balance between performance and efficiency. Therefore, we use 20 particles for our experiments.

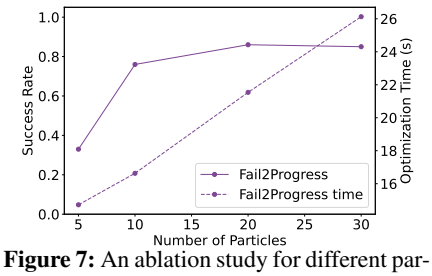

Figure 7: An ablation study for different particles.

A.6 Detailed Simulation Results

#Objs	Base	Original	Small	Large	Replanning	Sampling	Gradient	Fail2Progress
3	Points2Plans [2]	16%	18%	21%	28%	64%	52%	**90%**
3	Stow-GNN [3]	13%	15%	19%	24%	61%	51%	**83%**
3	Binary-Pred [4]	11%	13%	15%	23%	51%	47%	**78%**
5	Points2Plans [2]	10%	13%	15%	26%	54%	45%	**87%**
5	Stow-GNN [3]	9%	10%	14%	21%	52%	43%	**81%**
5	Binary-Pred [4]	8%	8%	13%	19%	43%	37%	**73%**
7	Points2Plans [2]	8%	9%	12%	18%	42%	39%	**82%**
7	Stow-GNN [3]	8%	7%	11%	17%	41%	37%	**76%**
7	Binary-Pred [4]	5%	7%	9%	14%	30%	29%	**64%**
Average	Points2Plans [2]	11%	13%	16%	24%	53%	45%	**86%**
Average	Stow-GNN [3]	10%	11%	15%	21%	51%	44%	**80%**
Average	Binary-Pred [4]	8%	9%	12%	19%	41%	38%	**72%**

Table 3: Simulation experiments for the **Hierarchical Tabletop Organization** task across different numbers of objects. The comparisons demonstrate that Fail2Progress outperforms baselines by a large margin.

We provide the detailed simulation results in Table 3. Through the comparison, we find that Fail2Progress outperforms all baselines with different numbers of objects using different base architectures.

A.7 Detailed Sim2Real Gap

We provide a detailed analysis of the Sim2Real gap for our work. The Sim2Real gap can be caused by the following reasons: (1): Perception gap: This includes differences in rendering and visualization between simulation and the real world (e.g., real-world perception is less accurate and noisier). In simulation, ground-truth object segmentation are readily available, whereas in the real world, obtaining accurate information is challenging, especially for partial views and cluttered scenes. (2): Controller mismatch: This arises due to differences in robot control between simulation and the real world. Real robots would have latency, compliance, and joint limit constraints, which are often not fully modeled in simulation. (3): Object geometry gap: Real-world objects vary in material, shape, and appearance, and often differ from their simulation twin. This discrepancy is particularly significant for deformable objects. (4): Physical modeling: The physical modeling in simulation can be inaccurate, which contributes to unrealistic physical interactions.

A.8 Extra Related Work

Solving long-horizon manipulation tasks remains a significant challenge in the community. Traditionally, task and motion planning [51] addresses this problem by separating high-level symbolic reasoning from low-level geometric reasoning. However, TAMP methods typically rely on explicit 3D object models [1, 72, 73, 51, 74] and symbolic operators with predefined effects [75, 72, 73, 51, 74, 76]. Alternatively, recent works propose to sequence learned skills to handle geometrically complex tasks, but these approaches are also limited to hand-crafted states [5, 49, 77, 78]. A more recent study [2] introduces a method to learn the effects of skills directly from partial-view point clouds, enabling robots to reason about real-world scenarios involving hard-to-predefine object interactions. However, none of the existing skill effect models [5, 49, 77, 78, 2] reason about or learn from failures after deployment. Our work is the first to leverage the failure cases to improve skill effect models, thereby minimizing future failures.

A.9 Relations Definition

We use both unary relations and binary relations in this paper.

We consider the following unary relations: (1) `Movability`: indicates whether a specific segment is movable (e.g., a table is not `movable`). (2) Drawer identification: specifies whether a segment is a drawer. (3): Drawer state: determines whether a drawer is `open` or `closed`.

We define the following binary relations: (1) Spatial relations: includes six spatial relationships- `left, right, front, behind, above, below`- defined following [4]. (2) `In-contact`: identifies whether two objects are in contact. Ground-truth labels for this relation are obtained directly from the IsaacGym [17] simulator. (3) `Boundary`: Specifies whether an object is on the boundary of a supporting surface (e.g., a table or shelf). We define `boundary`(A, B) as true if object A is above object B, the distance between A and the nearest boundary of B is less than a threshold $\epsilon_{boundary}$, and the dimensions of B exceed ϵ_{bottom}. In this paper, $\epsilon_{boundary}$ is set to 0.1m, and ϵ_{bottom} is set to 0.2m. (4) `Inside`: indicates whether an object is inside another (e.g., a container). We define `Inside`(A, B) as true if the bounding box of A is completely contained within the bounding box of B.

A.10 Skills Definition

We use the following skills in this paper.

Pick-and-place: This skill enables the robot to grasp an object and place it in a specific pose. If the grasp pose or placement pose is outside the robot's current reachable space, the robot will first move its base to make it reachable before executing the arm motions. The continuous parameter encodes the difference between the placement pose and the grasp pose.

Push: This skill allows the robot to push multiple objects. The robot first moves to a pre-push pose and then moves its end-effector along the push direction for a specific distance. The continuous parameter encodes both the push direction and push distance.

Open/Close Drawer: This skill enables the robot to open or close a drawer. The corresponding continuous parameters encode the distance and direction of the motion.

Notably, if failures occur with a newly introduced skill, a new skill effect model can be trained to handle that skill effectively. Due to the composability of the skill effect model, the planning can incorporate all the skills.

A.11 Details of Skill Effect Models

A.11.1 Introduction

Given an observation, O_t, at time t, represented as segmented point clouds, the skill effect model encodes O_t to a latent state X_t using Enc. Using a decoder, Dec, the latent state can be decoded to either geometric states like object poses or symbolic states such as inter-object relations, $\mathcal{R}$. Furthermore, the latent state, X_t, could also be propagated by a skill $\phi_t(a_t)$ with a dynamics model Dyn to predict the latent state X_{t+1} at the next time step. The predicted latent state X_{t+1} could also be decoded to predicted object poses or relations. To simplify, in this paper, we use $\gamma(\cdot)$ to represent the skill effect model composing the different components Enc, Dec, and Dyn as $\gamma(O_t,\phi_t,a_t) = Dec(Dyn(Enc(O_t),\phi_t(a_t)))$, that outputs the probabilities of different relations in $\mathcal{R}$, with $\Gamma(O_0,\phi_{1:H},a_{1:H}) = \gamma_H \circ \gamma_{H-1} \circ ... \circ \gamma_1$, representing a composition of skill effect for a skill sequence $\phi_{1:H}(a_{1:H})$.

A.11.2 Implementation Details

The input of a skill effect model (e.g., Points2Plans) is a segmented point cloud at timestep t, denoted as $O_t = \{O_t^0,...,O_t^n\}$, where n represents the number of segments.

Encoder: We utilize PointConv [79] as the Enc. The employed PointConv architecture consists of three set abstraction layers, each processing input point data and corresponding positional data to produce sampled positional data and feature data as output. Both the input and output positional data have three channels. The first abstract layer samples 128 points, with 8 neighbors per point determined using a bandwidth of 0.1. It employs an MLP with 6 input channels (3 for positions and 3 for features), 32 output channels, and a kernel size of 1. The second layer reduces the sample size to 16 points with 16 neighbors per point and uses a bandwidth of 0.2. This layer's MLP takes 35 input channels (3 for positions and 32 for features), and outputs 64 channels with a kernel size of 1. The third layer is a "group all" layer that generates 128-dimensional features per segment, using a bandwidth of 0.4, Its MLP has 67 input channels (3 for positions and 64 for features) and 128 output channels, with a kernel size of 1.

Specifically, the encoder (Enc) processes each segment to generate a corresponding point cloud feature, represented as $P_t^i = Enc(O_t^i)$, where each point cloud feature has 128 dimensions. Additionally, we use positional encoding in PyTorch [80] to assign a unique identifier to each object, represented as ID_i, which also has 128 dimensions. For each object, we concatenate the point cloud feature and positional encoding to form $X_t^i = P_t^i \oplus ID_i$, resulting in a feature vector with 256 dimensions. Consequently, the latent state is represented in an object-centric form as $X_t = \{X_t^0,...,X_t^n\}$, where each object's latent state contains 256 features.

Dynamics: The dynamics model (Dyn) takes as input the latent state X_t along with the corresponding skill and continuous parameter $\phi_1(a_1)$. Since ϕ_1 encodes discrete parameter identifying the object to manipulate, we use positional encoding to represent the manipulated object ID as ID_i, which has 128 features. For the continuous parameter a_1, we use a simple MLP MLP_{para} to encode a latent continuous parameter with 128 features. The MLP_{para} consists of two layers, each with 128 neurons, using ReLU as the activation function. As a result, each skill is represented as a latent state $AL_1 = MLP_{para}(a_1) \oplus ID_i$, where AL_1 has 256 features.

Once the latent skill AL_1 is obtained, we use a transformer-based dynamics model, Dyn. The transformer comprises 2 sub-encoder layers, 2 attention heads in the multi-head attention mechanism, and a dimensionality of 256 for the input and output. Given the latent state X_t and the corresponding latent skill AL_1, Dyn outputs the change in each latent state, represented as δX_t. The predicted new latent state is then computed as $X_{t+1} = X_t + \delta X_t$. For long-horizon planning, the dynamics model can be applied recurrently as $X_{t+H} = Dyn(X_t, AL_{1:H})$.

Decoder: The decoder (Dec) consists of three distinct modules: a position decoder Dec_p, a unary relation decoder Dec_u, and a binary relation decoder Dec_b. All decoders take the latent state (X_t) as input.

Position Decoder (Dec_p): The position decoder processes the predicted changes in each latent state (δX_t) and outputs the predicted changes in object positions (δp_t). Dec_p is a three-layer network, with each layer containing 64 neurons and ReLU as the activation function.

Unary relation decoder (Dec_u): The unary relation decoder takes the absolute latent state (X_t) as input and outputs unary object relations. Dec_u consists of two layers, each with 64 neurons, and uses Softmax as the activation function because unary relations are binary variables.

Binary relation decoder (Dec_b): The binary relation decoder takes pairwise latent states ((X_t^i, X_t^j)) as input and outputs pairwise object relations, defined as $Dec_b(X_t^i, X_t^j)$. Dec_b is a three-layer network, with each layer containing 64 neurons. Like Dec_u, Dec_b uses Softmax as the activation function since binary relations are also binary variables.

Training Details: We collect ground-truth data from the simulation at the current step, including point cloud observations (O_t), relations ($\mathcal{R}_t$), and position (p_t). Ground-truth data at the next step is collected after executing a robot skill, which includes point cloud observations (O_{t+1}), relations ($\mathcal{R}_{t+1}$), and position (p_{t+1}). To train the skill effect model using the simulation dataset ($\mathcal{D}$), we employ several loss functions:

Current step detection loss: Using the current step point cloud observation (O_t), the model (Γ) predicts current step relations ($\hat{\mathcal{R}}_t$). The current step detection loss is calculated as $L_{detection} = CE(\hat{\mathcal{R}}_t, R_t)$, where CE denotes the cross-entropy loss.

Latent space regularization loss: The mode encodes the observations (O_t, O_{t+1}) into the current step latent state (X_t) and the next step latent state (X_{t+1}). Using a skill, Γ predicts the next time step latent state (X'_{t+1}), where X'_{t+1} is derived from O_t and the skill while X_{t+1} derives from O_{t+1}. The regularization loss, calculated as the L2 norm, is $L_{regulization} = ||X_{t+1} - X'_{t+1}||_2^2$.

Position loss: Based on O_t and the applied skill, the model predicts the change in object positions (δp_t). The position loss compares the predicted position changes with the ground-truth position changes: $L_{pos} = b \cdot \sqrt{a \cdot ||\delta p_t - (p_{t+1} - p_t)||}$. Here, $a = 12$ and $b = 5$ are used to balance other loss terms, as defined in [2].

Prediction loss: To minimize the difference between predicted relations ($R'_{t+1} = Dec_b(X'_{t+1})$) and ground-truth relations (R_{t+1}) at the next time step, we compute the prediction loss as: $L_{prediction} = CE(R'_{t+1}, R_{t+1})$.

The total loss is the sum of all four terms: $L = L_{detection} + L_{regulization} + L_{pos} + L_{prediction}$. We train and fine-tune the skill effect model using the Adam optimizer with a learning rate of 1×10^{-4}. In this paper, we use 10 epochs for the pre-training and 200 epochs for the fine-tuning.

Planning Details: To achieve the goal relations ($\mathcal{G}$), we employ a shooting-based approach to sample the continuous parameters ($a_{1:H}$) given the initial observation (O_1) and the plan skeleton ($\phi_{1:H}$). To maximize the likelihood of achieving the goal relations, we sample a set of continuous parameters $\{a_{1:H}^j\}_{j=1}^{Ka}$ from the robot's workspace. Each continuous parameter sequence $a_{1:H}^j$ is rolled out, and we select the sequence that maximizes the probability of satisfying $\mathcal{G}$.

A.12 Real-to-sim details

For the real-to-sim process, SVI generates both simulation states and robot skills. The simulation states specify the pose of each object.

To create a simulation scene, we assume a set of object shape priors, including cuboids, open boxes, shelves, drawers, and tables. Based on the semantics of each segment in the observation, our method selects the appropriate object shape prior. The bounding box of each segment determines the dimension of the corresponding object in the simulation. By combining these dimensions with the object poses, we can construct the simulation scene.

For the robot skills, we directly execute the parameterized skills within the simulation, starting from the initial scene. During the process, we could record point clouds and object relations both before and after manipulation. Combined with the executed robot skills, we can generate a fine-tuning dataset to refine the skill effect model.

Note that we select this bounding-box approximation for the real-to-sim approach due to efficiency considerations. However, Fail2Progress can compliment other real-to-sim approaches [15, 14, 16].

A.13 Stein Update Details

A.13.1 Generating State Samples

First, we aim to solve for the simulation state set S^+. Here we want to find samples $q(S) = \{s_i^+\}_{i=1}^M$ that approximate the posterior distribution $P(r^F \mid O^+ = \xi(S)O^F)P(S)$, where $P(S)$ is a uniform prior over all feasible simulation states. The posterior distribution ensures that the transformed point clouds match the relations in the failure case r^F. This defines the following variational inference problem:

$$\underset{q(S)}{\operatorname{argmin}} D_{KL}\Big(q(S) \,\Big\|\, \Gamma(r^F \mid O^+ = \xi(S)O^F)P(S)\Big) \tag{4}$$

For each state particle s_i^+, the Stein update term is:

$$\Phi(s_i^+) = \frac{1}{M}\sum_{j=1}^{M}[k(s_j^+, s_i^+)\nabla_{s_j^+} \ln P(\mathcal{R}^F \mid \xi(s_j^+)O^F) + \nabla_{s_j^+} k(s_j^+, s_i^+)] \tag{5}$$

where $k(s_j^+, s_i^+)$ is a kernel function that defines the similarity between different particles. The first term in Eq. 5 represents an attractive force that pushes the particles to move in a direction based on the gradient while the second term is a repulsive term that prevents the particles from collapsing. This update can generate object states that match the failure case while ensuring diversity over object states.

A.13.2 Generating Action Samples

Given our state samples generated by using Stein variational inference to approximate the distribution in Eq. 3b, we can now turn our attention to solving for the action set A^+. To formulate this problem we make use of the generalized Bayesian inference framework outlined above. Here we define the loss function, $\mathcal{L}$, to be the entropy loss defined in Eq. 3a and let $\beta = 1$. Note that the variational distribution $q(A) = \{(s_i^+, a_i^+)\}_{i=1}^M$, however we keep the values of s_i^+ fixed and search only over actions. This defines the following variational inference problem:

$$\underset{q(A)}{\operatorname{argmin}} \mathbb{E}_{s^+, a^+ \in q(A)}\left[\prod_{r \in \mathcal{R}^F} -H(\Gamma(r \mid \xi(s^+)O^F, \phi^F, a^+, D))\right] + \quad D_{KL}\Big(A^+ \,\Big\|\, P(A)\Big) \tag{6}$$

where $P(A)$ is uniform prior over actions.

The Stein update term for the action particles a_i^+ is:

$$\Phi(a_i^+) = \frac{1}{M}\sum_{j=1}^{M}[k(a_j^+, a_i^+)\cdot\nabla_{a_j^+} \ln H(\Gamma(\mathcal{R}^F \mid \xi(s_j^+)O^F, \phi^F, a_j^+)) + \nabla_{a_j^+} k(a_j^+, a_i^+)] \tag{7}$$

A.13.3 Implementation Details of SVI

We use RBF kernels for SVI and follow previous works [19, 62] by applying the median heuristics to determine the kernel bandwidth. Additionally, the step size is optimized using the Adam optimizer [81].

A.14 Detailed Generalization Experiments

Generalization Evaluation: We assess the generalization capability of Fail2Progress compared to the Gradient and Sampling baselines in the **Multi-object Transport** task. First, we evaluate generalization to an unseen number of objects, as shown in Table 4. The model is fine-tuned only on scenarios with 3 objects and tested on unseen scenarios with 5 and 7 objects. While all approaches experience some performance degradation, Fail2Progress maintains strong performance, even in scenarios with 7 objects. In contrast, Gradient and Sampling perform poorly, particularly in the 7-object scenarios. Next, we assess generalization to unseen viewpoints, also shown in Table 4. Fail2Progress demonstrates robust performance across two unseen viewpoints and consistently outperforms Gradient and Sampling baselines. For both evaluations, we perform 100 trials per approach for each evaluation metric. Visualizations of these generalization scenarios are provided in Fig. 8.

Generalization Scenarios	3objs ↑	5objs ↑	7objs ↑	view1 ↑	view2 ↑
Fail2Progress	**87%**	**81%**	**71%**	**83%**	**85%**
Gradient	51%	40%	18%	42%	44%
Sampling	62%	45%	23%	51%	47%

Table 4: Generalization results for the **Multi-object Transport** task. We show the generalization capability of Fail2Progress with respect to different numbers of objects and different viewpoints (5objs, 7objs, view1, and view2 are unseen in the training dataset). Evaluations on unseen objects and unseen viewpoints show that Fail2Progress performs well and outperforms the best-performing baselines (Sampling and Gradient).

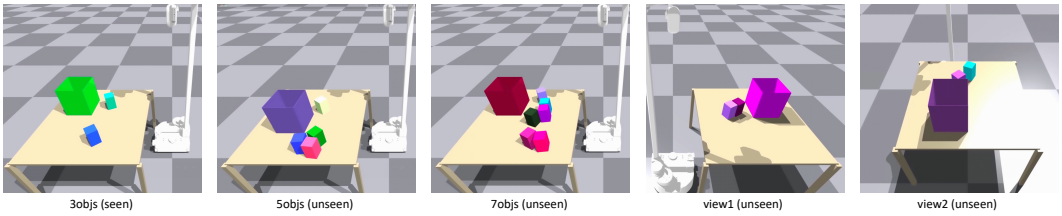

Figure 8: Visualizations of simulation generalization scenarios. Fail2Progress, fine-tuned on a dataset with 3 objects, successfully generalizes to scenes with 5 and 7 objects. Additionally, Fail2Progress demonstrates generalization to two unseen viewpoints.

A.15 Experimental Details

We now provide further details regarding the datasets. We first report the failure relations satisfaction score, which measures the percentage of relations in each dataset that match failure cases. These scores for Fail2Progress, Original, Small, Large, Sampling, and Gradient are **96.2%**, 82.4%, 82.9%, 82.8%, 89.7%, and 90.4%, respectively. Second, we report the standard deviation of the action parameters to evaluate the diversity of continuous action values. The corresponding standard deviations of the action parameters are **0.673m**, 0.415m, 0.421m, 0.424m, 0.523m, and 0.347m. These details support the findings presented in the main paper. The Original, Small, and Large baselines perform poorly because their training datasets do not capture out-of-distribution failures, resulting in lower failure satisfaction and action diversity scores. In contrast, Fail2Progress achieves the highest failure relation satisfaction and exhibits the most diverse actions, outperforming all baselines, including Gradient and Sampling. We attribute this superior performance to the ability of SVI to approximate high-dimensional, multi-modal posterior distributions. Note that we use rejection sampling as the sampling baseline, and it is significantly slower than Fail2Progress (Appx. A.3).

A.16 Additional Baseline for Domain Randomization

Domain randomization is a technique used to address failures arising from the Sim2Real gap. If one considers randomized objects, poses, and environments as domain randomization, then all of our Small and Large baselines can be considered domain randomization baselines.

Furthermore, we conduct experiments where actions are held constant while object states and environments are randomized to generate failure-targeted training data using Points2Plans architecture. The average success rate for this domain randomization is 48%, which is significantly lower than our proposed approach, Fail2Progress, which achieves an average success rate of 86%. The comparison details are shown in Table. 5. We conduct 100 trials for each approach and each object count.

	3objs ↑	5objs ↑	7objs ↑
Fail2Progress	**90%**	**87%**	**82%**
Domain randomization	59%	49%	36%

Table 5: Comparison against a domain randomization baseline using Points2Plans architecture.

A.17 Hardware Information

All the skill effect models are trained and fine-tuned on a standard workstation with an NVIDIA GeForce RTX 3090 Ti GPU. All the real-world experiments are conducted with a Stretch-re2 from Hello-Robot.