Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

PoCo: Policy Composition from and for Heterogeneous Robot Learning

Lirui Wang, Jialiang Zhao, Yilun Du, Edward H. Adelson, Russ Tedrake
MIT CSAIL
https://liruiw.github.io/policycomp

***Abstract*—Training general robotic policies from heterogeneous data for different tasks is a significant challenge. Existing robotic datasets vary in different modalities such as color, depth, tactile, and proprioceptive information, and collected in different domains such as simulation, real robots, and human videos. Current methods usually collect and pool all data from one domain to train a single policy to handle such heterogeneity in tasks and domains, which is prohibitively expensive and difficult. In this work, we present a flexible approach, dubbed Policy Composition, to combine information across such diverse modalities and domains for learning scene-level and task-level generalized manipulation skills, by composing different data distributions represented with diffusion models. Our method can use task-level composition for multi-task manipulation and be composed with analytic cost functions to adapt policy behaviors at inference time. We train our method on simulation, human, and real robot data and evaluate in tool-use tasks. The composed policy achieves robust and dexterous performance under varying scenes and tasks and outperforms baselines from a single data source and simple baselines that pool very heterogeneous data together in both simulation and real-world experiments.**

I. Introduction

Training generalist robotic policies with massive amounts of computational power and data using simple algorithms such as behavior cloning [1, 2] has gathered considerable interest recently [3, 4], inspired by the success of foundation models in fields such as natural language processing and computer vision [5, 6, 7]. Despite the growing size of robotic datasets [3], the dominant robot learning pipeline is still training specialist models with a single robot for a single task without any flexibility in behaviors [8, 9, 10]. One of the key challenges for robot imitation learning under multi-task and multi-domain data at scale is data heterogeneity: in addition to the vast amount of tasks that robots can be trained to perform, robotic datasets were usually built from different data sources, such as simulations, human demonstrations, and real-robot data with different modalities such as color images, depth images, and tactile data. In this paper, we will refer to settings that accept these varieties of modalities, embodiments and tasks as **heterogenous robot learning**. The efficient use of such data is not obvious due to such heterogeneity and most existing methods often only leverage one source of data such as real-world data [3] or simulation data [11, 12].

Similar to how humans learn complex manipulation behaviors that require common sense and reasoning [13] through mental simulation, watching, and active interaction, robot policy learning benefits from diverse sources of data. Simulation holds the promise to provide large-scale training data with massive diversity, yet policies trained with simulation usually suffer from the sim-to-real gaps [11]. Real-world human demonstrations are the largest existing data source and they are easy to acquire to teach high-level motions [14, 8]. Yet they suffer from embodiment gaps, especially on contact-rich motions. Robot teleop data has the least domain shifts and delivers impressive demos [9, 15], yet it can be expensive to acquire at large scales. Moreover, data of different modalities is often gathered with different hardware for each task. Given such massive heterogeneity in data, it remains unclear how to train policies on such data, that can achieve both dexterity and generalization to new scenes and tasks.

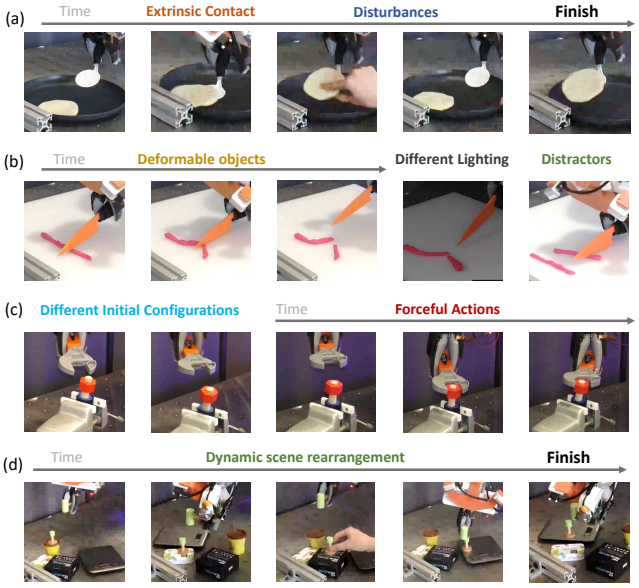

Fig. 1: **Generalizable Tool Use.** Policy composition can be applied to multiple tool-use tasks using a spatula, knife, wrench, and hammer. Resultant policies generalize across disturbances (a) and distractors (b) across different initial configurations and settings which require applied force (c) and dynamic rearrangements of scenes (d). The horizontal axis shows the time dimension for each executed trajectory.

We propose Policy Composition (PoCo), a framework (Figure 2) to **compose** multiple sources of data and objectives in different modalities and tasks with associated behaviors. This allows us to naturally divide and conquer the combined challenging problem with multiple policies and allow adaptation at test time. Specifically, each policy is instantiated probabilistically using a trajectory level diffusion model [16, 17, 9], enabling policies to be composed together

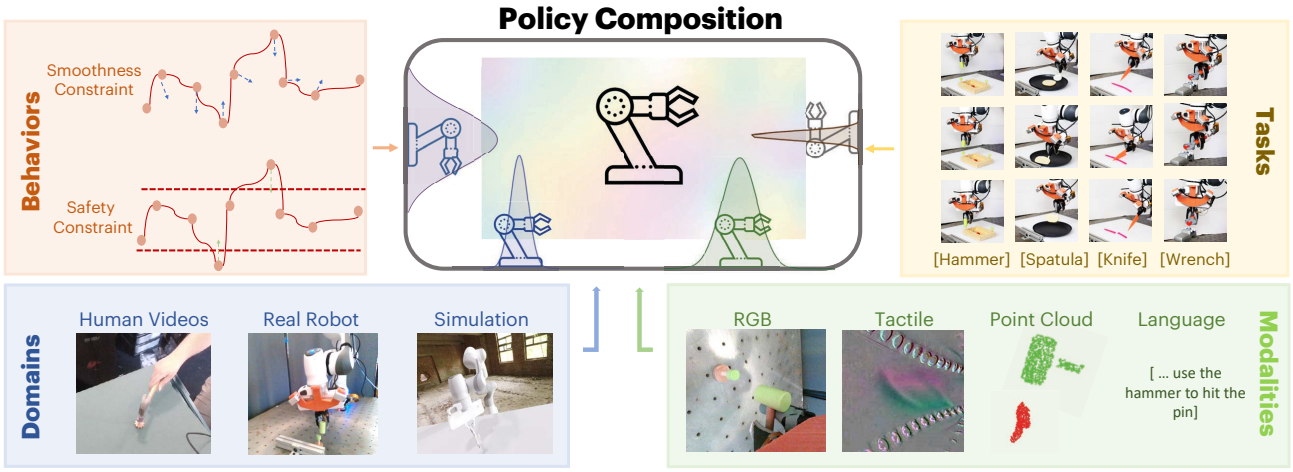

Fig. 2: **Policy Composition.** PoCo combines information across behaviors, tasks, modalities, and domains using probabilistic composition. This allows our approach to modularly combine information at prediction time (with no retraining), enabling generalization to challenging tool-use tasks, by leveraging information from multiple domains ().

by combining score predictions [18, 19, 20, 21, 22, 23]. In addition to representing cost functions [18], constraints [21, 24], and goals [19], learned probability distributions can represent policies operating across distinct sensor modalities from domains such as simulation, real robots, and human videos. We apply our methods for the tasks of **generalizable robotic tool-use** [25, 26, 27, 28] policies. We illustrate how different compositions of policies at the task, behavior, and policy levels enable robust generalization to novel scenes and tasks without fine-tuning.

In comparison to approaches to learning generalist policies through representation learning [29] or through large-scale data-pooling [3], our approach does not require extensive data engineering or data sharing to align observation and action spaces as individual policies are learned modularly on separate domains of data. Furthermore, our modular approach enables us to quickly adapt to out-of-distribution settings with additional sources of data or tasks with control [19, 20], by simply training additional policies without discarding information about prior tasks. In contrast to individually trained components (such as perception subsystem) in a robotic system [30], PoCo allows multiple instances of the policies trained independently to combined in arbitrary combinations in parallel to improve performance.

Via simulation testing on the Fleet-Tools benchmark [27], we demonstrate that the task composition improves diffusion policies in multi-task settings with various tools such as spatulas and wrenches. We further show that test-time behavior compositions can improve desired behavior objectives such as smoothness and workspace constraints. In the real world (Figure 1), we compose across policies trained with different modalities such as image, point cloud, and tactile image, and across domains such as in simulation and the real world. This allows the policies to stay robust across changes in manipulands, distractor objects, camera viewpoints and angles, as well as demonstrate dexterous retrying and fine-grained behaviors. We demonstrate that the overall compositions provide success rate improvements of 20% in both simulation and the real world compared to baselines.

The contributions are summarized as follows: (a) we propose PoCo, a policy composition framework to use probabilistic composition of diffusion models to combine information from different domains and modalities (b) we develop task-level, behavior-level, and domain-level prediction-time compositions for constructing complex composite policies without retraining (c) we illustrate the scene-level and task-level generalization of PoCo across simulation and real-world settings for 4 different tool-use tasks, and demonstrate robust and dexterous behaviors.

II. Related Works

A. Diffusion Models

Diffusion models [31, 32] are a class of probabilistic generative models that iteratively refine randomly sampled noise into draws of underlying distributions. Compared to other generative models such as Variational Autoencoder (VAE) [33] and Generative Adversarial Networks (GAN) [34], diffusion models, or more generally Energy-Based Models [35, 36], learn the gradient fields of an implicit function which can be optimized at inference time to readily compose with multiple models. Moreover, diffusion model composition has achieved impressive results in generating high-dimensional distributions such as images and videos [37, 38, 39].

Recently, diffusion models have been applied to robotic applications [40, 16, 17, 9], and enjoy rigorous theoretical guarantees [41]. These successes are attributable in part to the connection between diffusion processes and traditional gradient-based trajectory optimization, as practiced in robotic planning, optimal control, and reinforcement learning [42, 43, 44, 16]. The implicit function representation of diffusion models allows them to be composed with external probability distributions [16, 45], and researchers have had empirical successes in composing multiple diffusion models [39, 46, 22, 47, 23, 24].

B. Compositional Models in Robotics

In robotics, probabilistic composition has been previously explored as an approach for structured generalization [48, 16, 21, 24, 19, 49, 20, 18] at inference time. Previous works have also investigated a probabilistic view on composing energy-based reactive policies for robotic motion planning such as HiPBI [20]. In BESO[19], score functions for goal-conditioned imitation learning are composed using classifier-free guidance of an unconditional and conditional goal policies. In SREM and CCSP [21, 24], scene rearrangement is accomplished by composing EBMs to represent different subgoals in compositions of object scenes. In SE3 Diffusion Fields [18], learned cost functions are used as gradients for joint motion and grasping planning. Leveraging a similar score combination process, our work explores how such probabilistic composition can be applied to learned policies, enabling the combination of multidomain and multitask data to effectively solve complex tool-use tasks.

C. Multi-task and Multi-domain Imitation Learning

Multi-task learning and models that exhibit multi-task behaviors have shown impressive performance in computer vision [50, 51], natural language processing [52, 53]. Some of the most promising approaches in robot learning use direct data pooling, either from robot-only data such as RT series [54, 55, 56], or mixtures of human and simulation data. [8, 57, 58]. At the same time, there is a rich tradition of explicitly accounting for problem structure in multi-task learning [59], meta-learning [60, 61, 62], and few-shot learning domains [63]. Whereas data-pooling approaches require a careful balance of constituent data distributions, and approaches using problem structure [3] require additional algorithmic complication, our method composes diffusion policies learned from different domains at inference time with minimal algorithmic overhead.

D. Robotic Tool-Use

Tool-use is a widely studied problem for robotics [64, 65] and the AI community [66, 13, 67]. Compared to the well-studied grasping and planar-pushing problems, tool-use tasks require additional skills such as fine contact-rich dexterity, object-object affordance, and compositional generalization [13, 27]. To improve tool-use capability of robots, prior works have studied using model-based methods [64, 65], sparse perception features [68, 26], language conditioning [25, 69], simulation and online videos [70], point cloud inputs [28]. Due to the partial occlusion and the extrinsic contact sensing problems during tool-using, some works [71, 72, 73, 74] have studied using tactile sensors for tool manipulations. Our work proposes a novel multi-modal learning system for real robots, humans, and simulations in common tool-use tasks [27] with industrial and household applications.

III. Diffusion Models for Trajectory Generation

We study how to combine information across different tasks and domains such as simulation environments, human demonstration videos, on-robot data in combination to solve new tasks in new domains. Specifically, following [16, 9], we can parameterize a generative policy π on an action trajectory τ with a conditional distribution $\pi(\tau|o)$ given a history of observations, denoted as o, which belongs to a certain set of observation spaces, e.g. the spaces of point clouds, images, or images of tactile sensor deformations. Regardless of observation space, $\tau \in \mathbb{R}^{H\times d}$ corresponds to action trajectory at a horizon H and action dimension d. Because our action trajectories lie in the same space, even though observations do not, we will be able to compose across multiple different observation spaces as described in what follows.

Restricted to any given observation space $\mathcal{O}_k$, we represent a policy $\pi(\tau|o)$ with a diffusion model, where the trajectory τ is generated from the policy using an iterative denoising process. To generate a trajectory from the diffusion model, a noisy sample τ_T is initialized from Gaussian noise $\mathcal{N}(0, I)$. This sample τ_T is then iteratively denoised using a learned denoising network ϵ_θ as follows:

$$\tau^{t-1} = \tau^t - \gamma_t \epsilon_\theta(\tau^t, t \mid o) + \xi, \quad \xi \sim \mathcal{N}\big(0, \sigma_t^2 I\big), \tag{1}$$

for T total steps of denoising with specified per-timestep noise magnitudes σ_t and step size γ_t. The final generated sample τ_0 corresponds to the predicted trajectory of the policy π_θ

To train the denoising network ϵ_θ, we corrupt the trajectory τ across a randomly selected noise level $t \in \{1 \dots T\}$, and a corresponding corruption $\epsilon^t = \alpha^t \epsilon$ is added to a clean trajectory τ_0. The denoising model ϵ_θ is trained to predict the added noise to the trajectory using the MSE objective:

$$\mathcal{L}_{\text{MSE}} = \|\epsilon^t - \epsilon_\theta(\tau_0 + \epsilon^t, t \mid o)\|^2. \tag{2}$$

The denoising function ϵ_θ estimates the score $\nabla_\tau E(x)$ of an underlying EBM (unnormalized) probability distribution [75, 22], where $e^{-E(\tau)}$ represents the desired trajectory distribution $\pi(\tau \mid o)$, and it also allows diffusion models to be easily composed with other probability distributions.

IV. Learning Compositional Policies

In this section, we introduce the idea of compositional policy learning to handle the heterogeneous training and testing settings in robotics. For two probability distributions p_1, p_2, we compose them in the following form [76, 39, 23, 49, 20, 21, 18]

$$p_{\text{product}}(\tau) \propto p_1(\tau) p_2(\tau). \tag{3}$$

This product distribution combines the information contained in each distribution – for instance, when $p_1(\tau)$ and $p_2(\tau)$ represent Gaussian PDFs of the uncertainty from two sensor readings, the composed product Gaussian PDF $p_{\text{product}}(\tau)$ combines the mean and uncertainty estimates across both sensors. To expand this general composition idea in robotic policy learning with heterogeneous data and diffusion models, we first introduce how policies learned across data domains and modalities of tasks and objectives can be represented as probability distributions in Section IV-A. In Section IV-B, we discuss how we can compose different probability distributions in the form of a product distribution to leverage information across different modalities, domains, tasks, and behaviors. Finally, we discuss implementation details of how

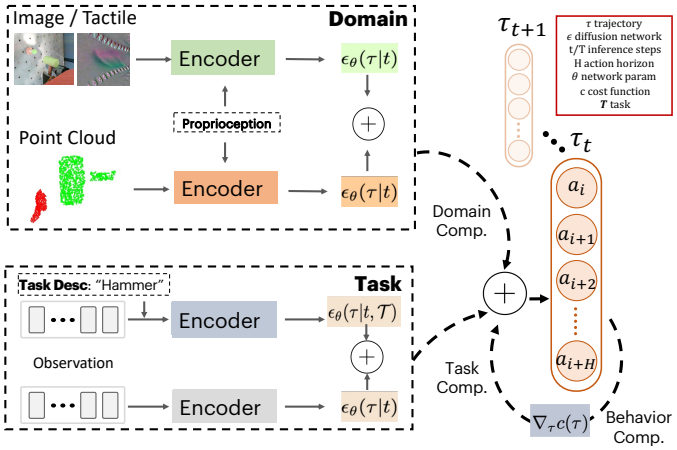

Fig. 3: **Illustration of Policy Composition.** Policies are composed at test time by combining gradient predictions. This can apply to *domain composition* by combining policies that train with different modalities such as image, point cloud, and tactile images. It can also be used with different tasks in *task composition* and additional cost functions for desired behavior with *behavior composition*. The only assumption is that the diffused outputs for each model need to be in the same space, i.e. the action dimension and action horizon. We denote the composition operator using a "plus" sign.

to sample from composed policies using diffusion models in Section IV-C.

A. A Probabilistic Perspective on Robotic Heterogeneity

We propose a probabilistic perspective [39, 24, 21, 49] to combine information across different domains such as simulation, humans, and real robots, across sensory modalities such as RGB images and point clouds, and across different behavioral constraints such as smoothness and collision avoidance. Training a generic policy directly across all these sources of information has been an outstanding challenge due to these heterogeneous modalities and objectives present in the data.

Let us first describe how we model these different forms of heterogeneity: domain, behavior, task, and sensing modality. We let *modality* $\mathcal{M} \in \{\mathcal{M}_{\text{tactile}}, \mathcal{M}_{\text{pointcloud}}, \mathcal{M}_{\text{image}}, \mathcal{M}_{\text{proprioceptive}}\}$ denote a sensing modality - tactile images, pointcloud, RGB image, or proprioceptive information including joint angles and end effector poses. We describe below how policies can take as input observations from any combination of these modalities.

Next, we consider multiple *data domains* $\mathcal{D}$ of demonstration trajectories. These include simulation data $\mathcal{D}_{\text{sim}}$, human video demonstrations $\mathcal{D}_{\text{human}}$, or on-robot teleoperated data saved from robot sensors $\mathcal{D}_{\text{robot}}$. In addition to different data distributions, different domains also have (a) different control frequencies, and (b) data from different combinations of modalities but crucially (c) *must have the same actions spaces* to enable composition in our current approach.

We then allow for the constraints on the desired behavior via a cost function $c(\tau)$ on the action trajectories, such as smoothness costs $c_{\text{smoothness}}(\tau)$ and safety costs $c_{\text{safety}}(\tau)$. Finally, we allow the specification of the robot task $\mathcal{T}$ via natural-language text command, such as $\mathcal{T}_{\text{hammer}}$: "hammering with a hammer" or $\mathcal{T}_{\text{spatula}}$: "scooping with a spatula".

We now propose to learn a separate probabilistic model $p_{\mathcal{D}}^{\mathcal{M}}(\cdot|c,\mathcal{T})$ on each tuple $(\mathcal{M},\mathcal{D},\mathcal{T},c)$ of (modality, domain, task, behavioral cost). We begin by training a policy $p_D^{\mathcal{M}}(\cdot \mid \mathcal{T})$ with no specified cost to exhibit a high likelihood on trajectories that are similar to expert demonstrations to achieve certain tasks $\mathcal{T}$ seen in domain $\mathcal{D}$ and modality $\mathcal{M}$. We instantiate this learned model as a denoising diffusion model. At inference time, we can incorporate the costs by sampling from the EBM distribution $p_D^{\mathcal{M}}(\tau \mid c,\mathcal{T}) \propto \exp(-c(\tau)) \cdot p_D^{\mathcal{M}}(\tau \mid \mathcal{T})$. In this setting, we choose to convert each cost probabilistically through the Boltzmann distribution $\exp(-c(\tau))$, although alternative probabilistic parameterizations are also valid as long as cost exhibits high-likelihood when the constraints are satisfied in a trajectory and low-likelihood when they are not. Concretely, our smoothness cost $c_{\text{smooth}}(\tau) = \|\tau''\|^2$ penalizes a discrete approximation of the second derivative of the trajectory; and workspace safety constraint cost $c_{\text{safety}}(\tau) = \|\min(\tau_{min} - \tau, 0)\|^2 + \|\max(\tau - \tau_{max}, 0)\|^2$, where τ_{min}, τ_{max} represent certain trajectory bounds.

Importantly, rather than *pooling all the data naively*, we train separate models for different (modality, domain, task) combinations and use compositionality, as described below, as a way to learn from this diverse and heterogeneous data.

B. Combining Information through Policy Composition

Given information across trajectories encoded by two probability distributions $p_{\mathcal{D}}^{\mathcal{M}}(\cdot|c,\mathcal{T})$ and $p_{\mathcal{D}'}^{\mathcal{M}'}(\cdot|c',\mathcal{T}')$, in policy composition, we propose to directly combine the information across both distributions at inference time by sampling from the product distribution. The key idea is to combine the score predictions from diffusion policies at inference time.

Under the notations defined in the previous section, we define

$$p_{\text{product}} \propto p_{\mathcal{D}}^{\mathcal{M}}(\cdot|c,\mathcal{T}) \cdot p_{\mathcal{D}'}^{\mathcal{M}'}(\cdot|c',\mathcal{T}'), \tag{4}$$

Note that we omit the observation $\mathcal{O}$ of the policy in p to further simplify the notation. This $p_{\text{product}}(\tau)$ will exhibit a high likelihood at all trajectories that are valid under both distributions, effectively encoding the information from both probability distributions. This approach allows each component neural network to be trained *individually*, and then *modularly* combined at test time, which further enables the incorporation of new sources of information.

Furthermore, as discussed in Appendix X-D this composition obtains the precise form of the conditional distribution given composed tasks and costs under the assumption that (1) there is mutual independence of tasks $\mathcal{T}$ and costs c and (2) conditional independence of tasks $\mathcal{T}$ and costs c given a trajectory τ. These assumptions hold when policies when different constraints and tasks are mutually independent. For instance, policy that is trained to mimic pancake lifting behavior and the costs of maintaining trajectory smoothness satisfy both assumptions. Below, we provide three examples of how policy composition can be used to improve policy performance.

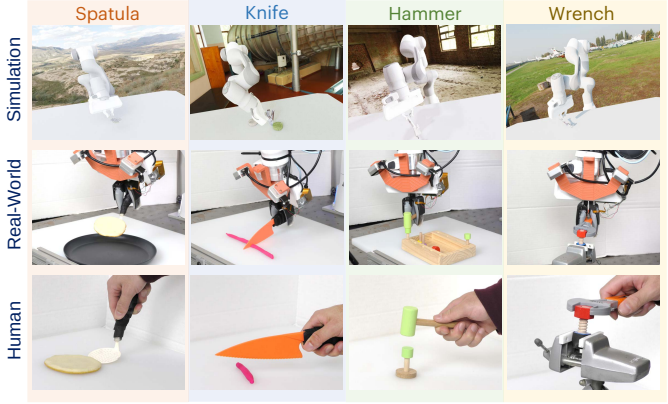

Fig. 4: **Task Visualizations.** Visualization of Tool-Use Tasks, "spatula, knife, hammer, wrench" are studied in this work in three different domains: simulation, real-world robots, and human demonstrations.

Task-level Composition. Given an unconditional distribution $p(\tau)$ describing possible trajectories for certain domains and modalities, and a conditional distribution $p(\tau|\mathcal{T})$ describing possible trajectories for a specified task $\mathcal{T}$, we can use composition to combine information across both distributions to synthesize a policy more likely to accomplish task $\mathcal{T}$.

In particular, we can construct the policy composition

$$p_{\text{product}}(\tau) \propto p(\tau)p(\mathcal{T}|\tau)^{\alpha} \propto p(\tau)\left(\frac{p(\tau|\mathcal{T})}{p(\tau)}\right)^{\alpha}, \quad (5)$$

for a chosen $\alpha > 1$. This particular policy composition puts extra weight on trajectories that are likely to accomplish task $\mathcal{T}$ (based off the weight α), improving the final quality of synthesized trajectories. This task-level composition procedure does not require separate training for each task, and trains a general policy that can achieve multiple task objectives, resembling a classifier-free diffusion model [46], which has also been applied to goal-conditioned policies [19] .

Behavior-level Composition. Given a distribution $p(\tau|c, \mathcal{T})$ of trajectories for a task $\mathcal{T}$ and a distribution $p_{\text{cost}}(\tau|c)$ encoding a cost function over a trajectory, we can combine distributions using

$$p_{\text{product}}(\tau) \propto p_{\text{cost}}(\tau)p(\tau|\mathcal{T}). \quad (6)$$

This policy composition combines information across the task distribution and the cost objective, ensuring that synthesized trajectories both accomplish a task and optimize a specified cost objective. This inference procedure is flexible and is similar to what a robotic trajectory optimizer will be able to do. In the action diffusion setting, the action trajectories need to be integrated to get the state trajectories.

Domain-level composition. Given two policy distributions $p_{D_1}(\tau|\mathcal{T}), p_{D_2}(\tau|\mathcal{T})$ representing information about valid trajectories to solve a task $\mathcal{T}$ trained on different sensor modalities and domains D_1, D_2, we can combine the information across both models together through

$$p_{\text{product}}(\tau) \propto p_{D_1}(\tau|\mathcal{T})p_{D_2}(\tau|\mathcal{T}). \quad (7)$$

This policy composition allows us to leverage information captured from different sensor modalities and domains, which is useful for complementing strengths of data gathered in one domain with another, *i.e.* when real-robot demonstrations are expensive to gather but more accurate, compared to simulation demonstrations which are cheaper to gather but not as accurate. We use feature concatenation for different modalities in the same domain for simplicity. Finally, note that multiple compositions may be nested together to further combine information across policies.

C. Implementing Compositional Sampling

One way of viewing the policy composition is through energy-based models (EBM). Given two EBM distributions [36] $p_1(\tau) \propto e^{-E_1(\tau)}$ and $p_2(\tau) \propto e^{-E_2(\tau)}$, the product of the two distributions $p_1(\tau)p_2(\tau)$ can be represented as a EBM $e^{-(E_1(\tau)+E_2(\tau))}$. We can sample from an EBM distribution $e^{-E(\tau)}$ using Langevin dynamics where given a sample initialized from τ^0 initialized from $\mathcal{N}(0, 1)$ we iteratively refine it using the gradient of the energy function

$$\tau^{t-1} = \tau^t - \gamma\nabla_\tau E(\tau) + \xi, \quad \xi \sim \mathcal{N}\left(0, \sigma_t^2 I\right), \quad (8)$$

where γ and σ_t are hyperparameters in realizing accurate sampling from the probability distribution. The Langevin sampling procedure in Equation 8 corresponds to the typical sampling procedure in diffusion models in Equation 1.

To combine policies, we compute $\nabla_\tau E_\theta(\tau)$ for each diffusion policy in each domain. Then, by using the score prediction of the denoising network $\epsilon_\theta(\tau, t)$ for each cost function, we can explicitly compute $\nabla_\tau c(\tau)$. This then allows us to use the standard diffusion sampling procedure in Equation 1 with the denoising network prediction substituted with the gradient prediction corresponding to composed distributions.

Concretely, this enables us to sample from task-level composition by using the sampling expression

$$\tau^{t-1} = \tau^t - \gamma_{\mathcal{T}}(\epsilon_\theta(\tau|t) + \alpha(\epsilon_\theta(\tau|t, \mathcal{T}) - \epsilon_\theta(\tau|t))) + \xi, \quad (9)$$

where $\xi \sim \mathcal{N}\left(0, \sigma_t^2 I\right)$. These two networks $\theta(\tau|t, \mathcal{T}), \theta(\tau|t)$ can be trained jointly with classifier-free training [46]. This enables us to sample from the behavior-level composition by using the sampling expression

$$\tau^{t-1} = \tau^t - \gamma_c(\epsilon_\theta(\tau|t, \mathcal{T}) + \nabla_\tau c(\tau)) + \xi, \quad (10)$$

where $\xi \sim \mathcal{N}\left(0, \sigma_t^2 I\right)$. Finally, assuming we have trained different networks θ_1, θ_2 for different domains $\mathcal{D}_1, \mathcal{D}_2$, we can sample from domain-level composition by using the sampling expression

$$\tau^{t-1} = \tau^t - \gamma_{\mathcal{D}}(\epsilon_\theta^1(\tau|t) + \epsilon_\theta^2(\tau|t)) + \xi, \quad (11)$$

where $\xi \sim \mathcal{N}\left(0, \sigma_t^2 I\right)$.

We note again that different compositions can use different observation inputs: for example, behavior composition does not rely on external observation, and domain composition can compose two separate policies using point cloud or image input respectively. In practice, we tuned the composition weights γ to demonstrate the composition behavior. Since we assume all models have shared the same action bounds for output normalization, the gradients (or noise prediction) with respect to the unnormalized trajectory (such as analytic costs) can be approximatedly combined with the normalized ones with a fixed linear transform and a scalar weight. It is also

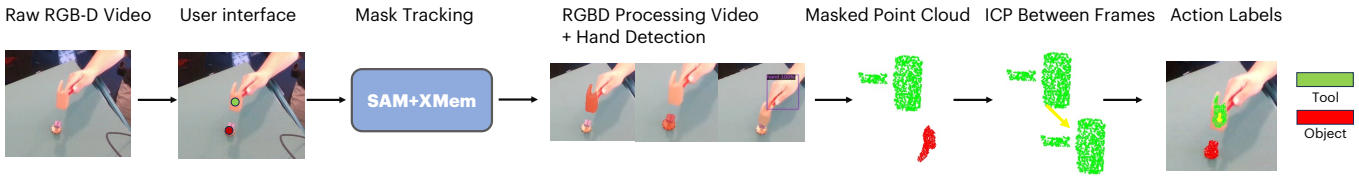

Fig. 5: **Dataset Processing Pipeline.** We show the human demonstration video collected in the wild from uncalibrated RGB-D cameras can be converted into a labeled trajectory (relative tool poses). The same pipeline applies to evaluating policy in the real world.

Algorithm 1 Sampling Process in Policy Composition

Input: Tasks $\{\mathcal{T}_k\}_{k=1}^{K}$, Policies Per Domain $\{\theta_i\}_{i=1}^{N}$, Costs $\{c_j(\tau)\}_{j=1}^{M}$, Base Multitask Policy ϕ.
Init: τ_T from Standard Gaussian
for Iteration $t = 1 \ldots, T$ **do**
  Initialize denoising step: $\nabla\tau^t = 0$
  for Task $k = 1 \ldots, K$ **do**
    Task Composition: $\nabla\tau^t = \nabla\tau^t + \epsilon_\phi(\tau|t) + \alpha(\epsilon_\phi(\tau|t, \mathcal{T}_k) - \epsilon_\phi(\tau|t))$
  for Behavior $j = 1 \ldots, M$ **do**
    Behavior Composition: $\nabla\tau^t = \nabla\tau^t + \gamma_c \nabla_\tau c_j(\tau)$,
  for Domain $i = 1 \ldots, N$ **do**
    Domain Composition: $\nabla\tau^t = \nabla\tau^t + \gamma_{\mathcal{D}} \epsilon_{\theta_i}(\tau|t)$
  Diffusion Step: $\tau^{t-1} = \tau^t - \gamma\nabla\tau^t + \xi$
Return: denoised trajectory τ_0

possible to use more advanced sampling techniques [23] and nested composition with an ensemble or mixture of policies for more complex behaviors. See Algorithm 1 and Figure 3 for a summary of the sampling process.

V. Implementations

We use a temporal U-Net structure with Denoising Diffusion Probabilistic Model (DDPM) [9, 16] at training time for 100 steps and Denoising Diffusion Implicit Model (DDIM) at testing time for 32 steps [9, 16]. Since the goal is to compose different diffusion models across domains $\mathcal{D}$ and tasks $\mathcal{T}$, we use the same action space for all models. For simplicity, a fixed normalization was used for the action bounds rather than dataset statistics, and the robot was controlled using end-effector velocity control. We focus on *generalizable tool-use tasks* to evaluate the proposed framework. The task is determined successful when certain thresholds are met. For example, a hammering task is deemed successful when the pin is hammered down. We use four task families (wrench, hammer, spatula, and wrench) for evaluation. We use ResNet-18 [77] as the image encoder and PointNet [78] as the point cloud encoder. See Figure 4 and Appendix X-E for more detailed discussion of different *domains*.

A. Simulation Setup

In the Fleet-Tools benchmark [27], we use the segmented tool and object point clouds as the agent observations, and the 6-DoF translation and rotation at the end-effector frame as the agent actions. The episode length is around 40 steps and the control frequency is 6Hz. Initial end-effector poses, initial object poses, and the tool-in-hand poses are randomized at each episode. In total, we collected around 50,000 simulated data points for each tool-object pair across 4 tool families. The demonstration trajectories in simulation are calculated with keypoint-based trajectory optimization. We apply data augmentation on both point clouds and actions in the training process. The test scenes are held out from the training process. We apply data augmentations that maintain the object-tool relationships. We also add point-wise noises, random cropping, and random dropping to the observed 512 tool and 512 object point clouds from the depth images and masks. Note that in the simulation experiment, only the point clouds are used for experiments (for their speed and ease of training). In the real-world experiments below, RGB-based policies are used for real-world collected data. See Appendix X-B for more details.

B. Real Robot Setup

We mount a wrist camera (Intel D405) and an overhead camera (Intel D435) to obtain local and global views of the scene. The near view range of D405 is useful for capturing the tool point clouds from the wrist view. We used a simplified version of the GelSight Svelte Hand [74, 79] installed on a Franka Emika Panda robot to handle different tools. One of the three GelSight Svelte tactile fingers is activated, which provides rich tactile information on tool poses, tool shape, as well as tool-object contacts. For teleoperation, we use an Oculus Quest Pro to track joystick poses, which are mapped to the velocity of the end-effector as action labels for imitation learning. The episode lengths of real-world teleoperation vary from 20 steps to 100 steps. The real robot policy frequency is 10 Hz when using RGB inputs only, or 6 Hz when segmented point clouds are used. See Appendix X-E for more details.

C. Human Demonstration Setup

The demonstration videos can be collected from uncalibrated cameras in the wild. The videos can then be labeled by a few clicks for only the first frame in each trajectory. We then use pre-trained segment-anything [5] for segmentation, XMem [80] for tracking, and hand detection pipelines [81] to extract an approximate hand pose, as shown in Figure 5. Point clouds of the tool and the object are extracted and transformed to the hand frame. Then we use the Iterative Closest Point (ICP) method to solve for the relative movements of the tool between two timesteps, which are used as action labels. Trajectory lengths vary from 20 steps to 100 steps. See Appendix X-E for more implementation details. Additionally, we plan to open-source all collected datasets for future research.

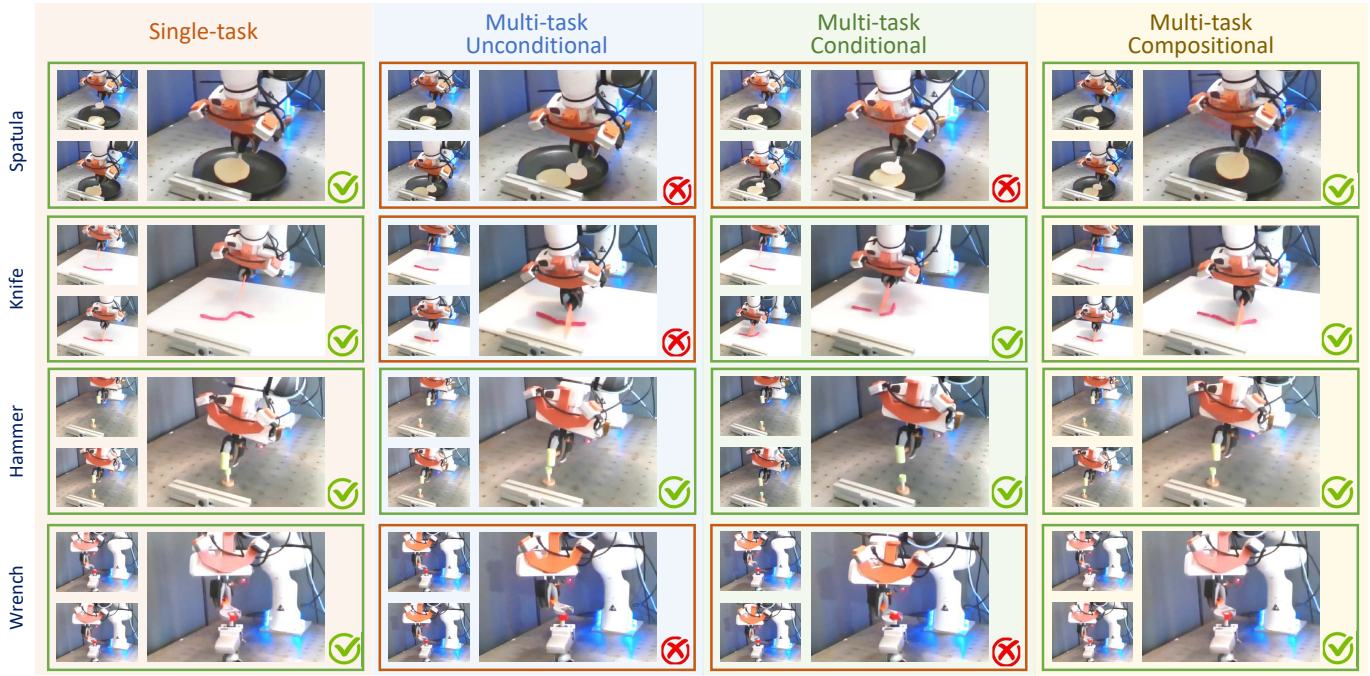

Fig. 6: **Task-level Qualitative Results.** By composing unconditional and task-conditional models, our composed policy can accomplish a diverse set of tasks and outperform unconditioned multitask policies.

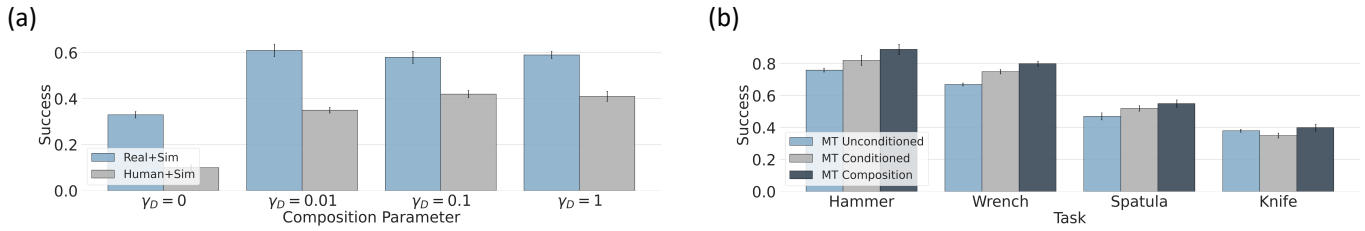

Fig. 7: **Effect of Policy Composition in Simulation. (a)** Task Composition performs the best in multitask policy evaluation in simulation. **(b)** Simulation policies help with composition across domains and evaluation in simulation.

Metric	**Success** ↑	**Smoothness** ↓	**Workspace** ↓
Normal	0.70	0.027	0.030
+Smoothness	0.67	**0.016**	0.038
+Workspace	0.67	0.019	**0.022**

TABLE I: **Effect of Behavior Composition.** By combining costs probabilistically, we can optimize metrics from each cost objective.

VI. Simulation Experiments

In this section, we explore the following question: when the test distribution is different from any single one of the training distributions, how each compositionality as described in Section IV can help adjust policy behaviors (VI-A), generalize across multiple tasks (VI-B) and multiple domains (VI-C) without retraining? In simulated experiments, we use the masked tool-object point clouds as the policy input $\mathcal{O}$. $\mathcal{O}$ is further encoded with a PointNet [78] before being fed into the diffusion model ϵ_θ. For each task, we perform 10 independent runs of each experiment on 50 scenes, and the average success rates together with its success criteria are reported. The model size and dataset size are the same for each domain in each experiment across baselines. The simulation environment is illustrated in Figure 8 and is used to evaluate both policies trained from real world data and simulation data.

A. Behavior-level Composition

In Table I, we use test-time inference to compose behaviors such as smoothness and workspace constraints. Recall that the smoothness costs c_{smooth} measure the squared norm of the averaged accelerations of the rollout trajectories, where workspace costs c_{safety} measure the squared norm of the trajectory outside of the workspace bounds. For each environment timestep, we integrate the action trajectories into the workspace pose trajectories and then compute the smoothness and collision costs, and compute its gradients with pytorch autograd for composition. We fix the composition weight to be $\gamma_c = 0.1$ in this setting. This maps to a combined diffusion step with the analytic cost functions $\epsilon^t = \epsilon_\theta(\tau) + \gamma_c c(\tau)$ in each step of the diffusion process. In Table I, we demonstrated the gains in smoothness and safety constraints by adding prior information to the action trajectories in the inference process. This composition is analogous to trajectory optimization widely used in robotic motion planning.

Setting	Human	Simulation	Real-Robot	Composition (Human+Real)	Composition (Sim+Real)
Vary Object Pose	1/5	5/5	2/5	4/5	5/5
Vary Robot and Tool Pose	1/5	5/5	4/5	5/5	5/5
Distractor	0/5	5/5	2/5	3/5	5/5
Novel Instance	1/5	3/5	2/5	1/5	5/5
Total	$15 \pm 2.2\%$	$90 \pm 4.3\%$	$50 \pm 4.3\%$	$65 \pm 7.4\%$	$100 \pm 0\%$

TABLE II: **Quantitative Results on Domain Composition.** Policy Composition improves average success rates compared to individual constituent policy across different scenes on four separate generalization axes, evaluated on the hammering task.

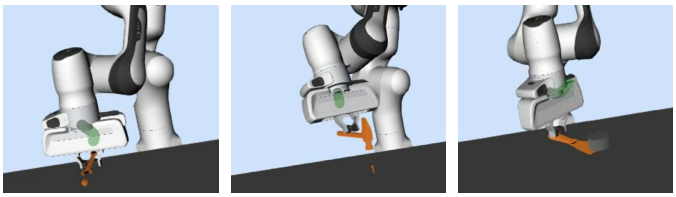

Fig. 8: **Simulation Qualitative Results.** We show some visualization with the inferred trajectory (in green) in the Drake simulation environments. These are tool-use tasks using a wrench, hammer, and spatula (left, middle, right). The tools are welded to the end effector.

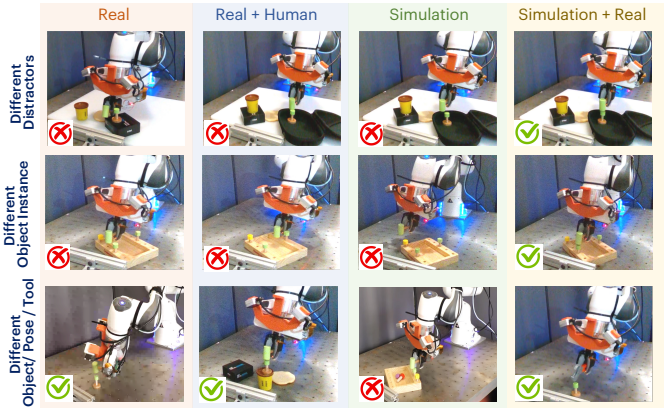

Fig. 9: **Domain Composition Comparison.** By composing policies trained in simulation, human, and real data, our approach can generalize across multiple distractors (first row), with varying object and tool poses (second row), and across new object and tool instances (third row).

B. Task-level Composition

In Figure 7 (b), we show that the task composition of conditional and unconditional multi-task tool-use policies outperforms both unconditional and task conditional multi-task tool-use diffusion policies. To construct a conditional multi-task tool-use policy, we assign each tool-use task a text name, such as "wrench", and train a conditional tool-use policy based off this text by using a T5 encoder [82] corresponding to learning $\epsilon_\theta(\tau|\mathcal{T})$. To construct an unconditional multi-task tool-use policy, we dropout text labels with a frequency 0.1 when training our conditional multi-task tool-use policy and replace the text embedding with a fixed latent vector corresponding to learning $\epsilon_\theta(\tau)$. Our task-level composition then corresponds to $(\epsilon_\theta(\tau) + \alpha(\epsilon_\theta(\tau|\mathcal{T}) - \epsilon_\theta(\tau))$, which can also be seen as classifier-free guidance [46]. From Equation 5 and its gradient form 9, we see that when the task weight $\alpha = 0$, this policy maps to unconditioned multitask policies, when $\alpha = 1$, it maps to standard task-conditioned policies, and when $0 < \alpha < 1$, we are interpolating between task conditioned and task unconditional policies. When $\alpha > 1$, we can get trajectories that are more task-conditioned. We fix the composition hyperparameter $\alpha = 1.5$ in Figure 7 and illustrate how such an approach improves performance over both unconditional and conditional policies.

C. Domain-level Compositions

In Figure 7 (a), we trained separate policy models for the simulation dataset θ_{sim}, human dataset θ_{human}, and robot dataset θ_{robot} using point cloud inputs and we evaluated domain-level composition in the simulation settings, i.e. real-to-sim evaluation when using θ_{human} or θ_{robot}. To construct a policy that can work well across domains, we use domain-level composition to fuse them and evaluate them in the simulation setting. Note that there is no train/test domain gap for θ_{sim}, which therefore performed well and can achieve 0.92 success rates. For example, in domains such as human data, we observe that composing with a more performant policy θ_{sim} with (e.g. $\gamma_{\mathcal{D}} = 0.1$) improves the performance drastically ($\gamma_{\mathcal{D}} = 0$) in simulation. This domain level composition maps to a combined diffusion step from both policies $\epsilon^t = \epsilon_{\theta_{\text{human}}} + \gamma_{\mathcal{D}}\epsilon_{\theta_{\text{sim}}}$ in each step of the diffusion process.

VII. Real-world Experiments

In this section, we experiment with PoCo in real-world experiments on tool-use tasks to demonstrate how data from different domains and tasks can be composed to improve generalization performance. The experiments are conducted in the same setting for each test episode across all methods.

A. Scene-level Generalization

In this section, we experiment with composing the trained policies conditioned on observations $\mathcal{O}_{\text{realworld}}$ as RGB images $\mathcal{M}_{\text{RGB}}$ and tactile images $\mathcal{M}_{\text{tactile}}$ in the real world, policies conditioned on observation $\mathcal{O}_{\text{human}}$ with point clouds $\mathcal{M}_{\text{pointcloud}}$ for human datasets, and policies conditioned on observation $\mathcal{O}_{\text{sim}}$ with point clouds $\mathcal{M}_{\text{pointcloud}}$ point clouds in simulation datasets, and in real-robot datasets. We focus on the reaching task which aims to reach a pin with a hammer.

In Figure 9 and Figure 12, we consider four common generalization axes across the scene level in robotics in this experiment: varying object poses, varying robot initial pose, varying tool poses, adding distractor objects, and replacing the object with novel instances from the same classes. We note that the performance of simulation-trained point-cloud

Train/Test	Spatula	Knife	Hammer	Wrench	Avg
Single-Task	8/10	8/10	5/10	5/10	$65\% \pm 0.5\%$
MT Unconditioned	6/10	5/10	5/10	4/10	$50\% \pm 0.6\%$
MT Conditioned	8/10	5/10	6/10	2/10	$53\% \pm 0.6\%$
MT Composition ($\alpha = 0.1$)	7/10	4/10	7/10	4/10	$55\% \pm 0.6\%$
MT Composition ($\alpha = 2$)	8/10	4/10	7/10	5/10	$60\% \pm 0.6\%$

TABLE III: **Policy Performance on Different Tool-use Tasks.** We compare among different ways to handle multitask (MT) diffusion policy training, and find that task composition overall leads to improved performance across tool-use tasks.

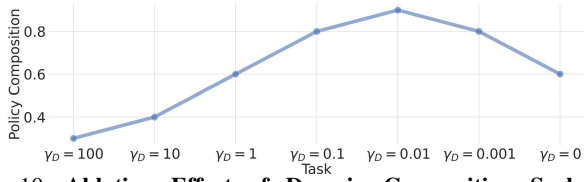

Fig. 10: **Ablation Effect of Domain Composition Scale.** We illustrate the effect of performance on the hammering task as we increase the composition coefficient between policies.

policies is inherently dependent on the image segmentation performance and point cloud quality. In Table II, we observe that the simulation policies are robust yet additional composition with real-world policy can help when one fails under certain circumstances. Interestingly, we observe that while human data trained policies and real-robot-trained policies are not performant under different scene challenges, their compositions can outperform each individual constituent.

B. Task-level Generalization

We evaluate robot policy performance on four different tool-use tasks in the real world. We call this multitask learning setting task-level generalization. In this experiment, all policies are trained with real-world RGB and tactile data with end effector poses as input. The multitask-conditioned policies use language features as task-specific features to concatenate with the observation features. The task-composition policies use language-conditioned classifier-free training. In Table III, we found that multitask policies can perform on par with task-specific policies conditioned on $\mathcal{T}_{\text{spatula}}$ and $\mathcal{T}_{\text{hammer}}$ for example. They both showed stability in fine actions in dexterous tasks and we expect scaling up to more tasks will further improve the policy representation. We also found that language-conditioned classifier-free training performs better than naively concatenating the features when training multitask tool-use policies. We found that the composition hyperparameter needs to stay within a range to be effective and stable. Surprisingly, the trained policies exhibit robust retrying behaviors toward disturbances and robustness towards changing lighting conditions and shadows, as shown in Figure 1.

C. Ablation Study

In this section, an ablation study is performed with several components of the proposed framework. We first compare with a baseline that naively pools all data from heterogeneous domains together for learning. The policy is trained on both simulation point cloud datasets and real-world image datasets for the hammering task, by mapping both modalities into a shared latent space for the policy. As shown in Table IV, this

Ablation	Data Pooling	No Tactile	No Rollout
Relative Success	-75%	-38%	-24%

TABLE IV: **Ablation of Real World Execution.** We analyze the effect of composition across modalities (data pooling), the use of tactile signals (no tactile) and the effect of predicting open-loop trajectory rollouts (no rollout) in the real world. The numbers represent the relative scale of the ablated method compared to the default method that involves policy composition from simulation and real world, tactile feedback, and closed-loop predictions.

naive approach performs poorly despite having larger model size, resulting in a 75% performance drop. We hypothesize that the constraints and the domain gaps between the two modalities cause challenges in leveraging both modalities efficiently without a vast amount of data. In addition, we find the tactile information to be important in certain tasks that require larger gripping forces and extrinsic contact interactions such as wrench using. Similar to [9], we also found rolling out action trajectory predictions (e.g. 4 steps) to both improve motion smoothness and task success rates. Finally, we ablated on the composition scale (Table IV) and found a suitable range for best performance beyond unconditional ($\alpha = 0$) and conditional ($\alpha = 1$) models.

VIII. Limitations and Future Work

Our system consists of several limitations and weaknesses. In this paper, we have proposed a simple form of policy composition where multiple policy distributions are multiplied together. While such a composition is effective, it does not enable temporal composition across long-horizon tasks and it assumes policies are roughly aligned in action horizon, scale, and frequency. As a result, our paper only shows the efficacy of PoCo on short-horizon tool-use tasks, and we believe extending PoCo to temporal composition is a rich direction for future work.

In addition, the policies can overfit and perform poorly in tasks that require delicate contact control. Some failure cases are illustrated in Appendix Figure 16. We believe some of the failure cases can be potentially improved by (1) increased data quantity, especially higher data coverage of corner cases or failure recovery cases, and (2) reduced computational costs at the inference time, such that more individual models can be composed and/or the composed policy can be executed at a higher frequency. The current policy composition can only run at 5Hz when combining two policies, and will linearly increase when adding more policies. There are also limitations when transferring the policy trained on one tool to another tool, even in the same class. Previous works [83, 84, 85] have several ideas that can be applied to address these tool-transfer constraints as well.

Finally, the three levels of compositions in our work are only a first attempt in this direction: for task composition, we only consider 4 tasks in the multitask domains, and extending to many more tasks would be interesting. For behavior composition, we only consider analytic cost functions in the simulation environments, and adding more interesting cost functions in the real world will be interesting. For domain-level composition, we only consider normal policy models

trained with in-house data, and extend to more diverse models trained with RT-X [3] and Ego4d [86] for example would be interesting.

IX. Conclusion

In this work, we proposed PoCo, a diffusion-based framework to compose robotic policies that tackle the data heterogeneity and task diversity problems in robotic tool-using settings. Without any retraining, our framework can flexibly compose policies trained with data from different domains (tactile images, point clouds, and RGB images), for different tool-use tasks (scooping, hammering, cutting, and turning), with different behavior constraints (smoothness and workspace constraints). In both simulation and the real world, we show that PoCo performed robustly on tool-using tasks and it showed high generalizability to various settings, compared to methods trained on a single domain. Future directions include temporal trajectory composition of different frequencies for long-horizon tasks, policy composition methods on large-scale datasets, and policy distillation from composed policies.

Acknowledgment

Lirui Wang is supported in part by Amazon Greater Boston Tech Initiative and Amazon PO No. 2D-06310236 and Defense Science & Technology Agency, DST00OECI20300823. Yilun Du is supported by NSF Graduate Fellowship. Jialiang Zhao is partially supported by Toyota Research Institute and Amazon Science Hub. The authors would also like to thank Sandra Q. Liu for her help in the fabrication of the hand used in this work. We thank MIT Supercloud for providing computing resources. The authors would like to thank Max Simchowitz, Kaiqing Zhang, and William Shen for reviewing an earlier version of this draft.

References

[1] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[2] J Andrew Bagnell. An invitation to imitation. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 2015.

[3] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. https://robotics-transformer-x.github.io, 2023.

[4] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[8] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[9] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[10] Lirui Wang, Yu Xiang, Wei Yang, Arsalan Mousavian, and Dieter Fox. Goal-auxiliary actor-critic for 6d robotic grasping with point clouds. In *Conference on Robot Learning*, pages 70–80. PMLR, 2022.

[11] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[12] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[13] François Osiurak and Arnaud Badets. Tool use and affordance: Manipulation-based versus reasoning-based approaches. *Psychological review*, 123(5):534, 2016.

[14] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

[15] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705. https://tonyzhaozh.github.io/aloha/*, 2023.

[16] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[17] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[18] Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through

diffusion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5923–5930. IEEE, 2023.

[19] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[20] Kay Hansel, Julen Urain, Jan Peters, and Georgia Chalvatzaki. Hierarchical policy blending as inference for reactive robot control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10181–10188. IEEE, 2023.

[21] Nikolaos Gkanatsios, Ayush Jain, Zhou Xian, Yunchu Zhang, Christopher Atkeson, and Katerina Fragkiadaki. Energy-based models as zero-shot planners for compositional scene rearrangement. *arXiv preprint arXiv:2304.14391*, 2023.

[22] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. *arXiv preprint arXiv:2206.01714*, 2022.

[23] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023.

[24] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional diffusion-based continuous constraint solvers. *arXiv preprint arXiv:2309.00966*, 2023.

[25] Allen Z Ren, Bharat Govil, Tsung-Yen Yang, Karthik R Narasimhan, and Anirudha Majumdar. Leveraging language for accelerated learning of tool manipulation. In *Conference on Robot Learning*, pages 1531–1541. PMLR, 2023.

[26] Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7278–7285. IEEE, 2020.

[27] Lirui Wang, Kaiqing Zhang, Allan Zhou, Max Simchowitz, and Russ Tedrake. Fleet policy learning via weight merging and an application to robotic tool-use, 2023.

[28] Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In *Conference on Robot Learning*, pages 1038–1049. PMLR, 2023.

[29] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[30] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[31] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[32] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[33] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[34] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[35] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[36] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[37] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[38] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[39] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33:6637–6647, 2020.

[40] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023.

[41] Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[42] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056, 2009.

[43] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The In-*

ternational Journal of Robotics Research, 37(11):1319–1340, 2018.

[44] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[45] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

[46] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[47] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. *Advances in Neural Information Processing Systems*, 34:13497–13510, 2021.

[48] Yilun Du, Toru Lin, and Igor Mordatch. Model based planning with energy based models. In *Conference on Robot Learning*, 2019.

[49] Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

[50] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018.

[51] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[52] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[53] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[54] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[55] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[56] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023.

[57] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*, 2023.

[58] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, pages 654–665. PMLR, 2023.

[59] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[60] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

[61] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[62] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[63] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[64] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. 2018.

[65] Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. Force-and-motion constrained planning for tool use. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7409–7416. IEEE, 2019.

[66] Joanna H Wimpenny, Alex AS Weir, Lisa Clayton, Christian Rutz, and Alex Kacelnik. Cognitive processes associated with sequential tool use in new caledonian crows. *PLoS One*, 4(8):e6471, 2009.

[67] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47):29302–29310, 2020.

[68] Pawel Gajewski, Paulo Ferreira, Georg Bartels, Chaozheng Wang, Frank Guerin, Bipin Indurkhya, Michael Beetz, and Bartłomiej Śniezyński. Adapting everyday manipulation skills to varied scenarios. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1345–1351. IEEE, 2019.

[69] Mengdi Xu, Peide Huang, Wenhao Yu, Shiqi Liu, Xilun Zhang, Yaru Niu, Tingnan Zhang, Fei Xia, Jie Tan, and Ding Zhao. Creative robot tool use with large language models. *arXiv preprint arXiv:2310.13065*, 2023.

[70] Kateryna Zorina, Justin Carpentier, Josef Sivic, and Vladimír Petrík. Learning to manipulate tools by aligning simulation to video demonstration. *IEEE Robotics and Automation Letters*, 7(1):438–445, 2021.

[71] Andrea Sipos and Nima Fazeli. Multiscope: Disambiguating in-hand object poses with proprioception and

tactile feedback. *arXiv preprint arXiv:2305.14204*, 2023.

[72] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3583–3590. IEEE, 2022.

[73] HJ Terry Suh, Naveen Kuppuswamy, Tao Pang, Paul Mitiguy, Alex Alspach, and Russ Tedrake. Seed: Series elastic end effectors in 6d for visuotactile tool use. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4684–4691. IEEE, 2022.

[74] Jialiang Zhao and Edward H Adelson. Gelsight svelte: A human finger-shaped single-camera tactile robot finger with large sensing coverage and proprioceptive sensing. pages 8979–8984, 2023.

[75] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[76] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[78] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[79] Jialiang Zhao and Edward H Adelson. Gelsight svelte hand: A three-finger, two-dof, tactile-rich, low-cost robot hand for dexterous manipulation. *arXiv preprint arXiv:2309.10886*, 2023.

[80] Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, pages 640–658. Springer, 2022.

[81] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9869–9878, 2020.

[82] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[83] Sascha Brandi, Oliver Kroemer, and Jan Peters. Generalizing pouring actions between objects using warped parameters. In *2014 IEEE-RAS international conference on humanoid robots*, pages 616–621. IEEE, 2014.

[84] Ulrich Hillenbrand and Maximo A Roa. Transferring functional grasps through contact warping and local replanning. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2963–2970. IEEE, 2012.

[85] Skye Thompson, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Shape-based transfer of generic skills. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5996–6002. IEEE, 2021.

[86] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[87] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[88] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[89] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

X. Appendix

In the Appendix Section X-A, we discuss the task descriptions and implementations of the tool-use tasks. In Section X-B, we discuss the method implementations for training and neural networks. In Section X-C, we show additional experiments for comparing neural network architectures. In Section X-D, we derive the formulate for the policy composition. Section X-E, we discuss simulation, human demonstration, and real-world robot setups. In Section X-F and Section VIII, we discuss data heterogeneity in robotics and some limitations of the proposed method.

A. Task Descriptions and Implementation

Although our framework can be applied to any policy setting or robotic applications, we choose 4 tasks in this work following [27]: `use a spanner to apply wrenches to a screw`, `use a hammer to hit a pin`, `use a spatula to lift a pancake from a pan`, and `use a knife to split a play-doh`. These tasks provide a balance of common robotic tasks that require precision, dexterity, and generality in object-object affordance. These tasks include features such as deformable object manipulation, extrinsic contacts, and forceful manipulation. The wrench task is considered a success if the nut is turned by 15 degrees. The hammer task is considered a success if the pin is reached by the hammer. The spatula task is considered a success if the object is lifted by the spatula. The knife task is considered a success if the Play-Doh is separated in half. The hammering task is considered successful if the pin is hammered down by 2cm. These tasks involve rich force and fine contact interactions as well as involve rigid, articulated, and deformable objects. While the hammering task is different from dynamic hammering tasks that human do, it still requires precise dexterous manipulation to execute. The real-world toy tool sets can be purchased through these links 1,2.

In each of the simulation task initialization, the scene can have randomized object pose, tool-in-hand pose, and robot initial joint angles. The hammer task is considered a success if the pin is pressed down 2 cm. The wrench task is considered a success if the wrench is rotated clock-wise by more than 0.4 radians. The spatula task is considered a success if the object is lifted by over 5 cm. The knife task is considered a success if the two objects are split by over 10 cm. See Figure 7 for successful task demonstrations.

B. Implementation Details

To implement each policy, all image inputs, including overhead camera, wrist camera, and tactile cameras are all processed with ResNet-18 [77]. We apply standard color jittering and motion blurring for real-world RGB inputs. For point cloud inputs, the point clouds are transformed in the end effector frame before computing the actions. We use masked point clouds for the tool (mask 1) and the object (mask 0) and feed into a standard PointNet to compute the features. Each encoder in each of the policies is trained separately from other datasets. We experimented with more advanced network architectures but did not find major improvements. We apply data augmentations that maintain the object-tool relationships. For example, we change the global frames for the end effector to mimic different ways of holding the tools. We also apply noises to either the object or tool point clouds and apply corrective actions to the action labels. We also add point-wise noises, random cropping, and random dropping to the observed 512 tool and 512 object point clouds from the depth images and masks. Different from the original diffusion policy implementation [9], we choose to use a fixed normalization for the action bounds rather than dataset statistics and use end-effector velocity control with six degrees of freedom parametrized in xyz and roll, yaw, pitch. The main reason is that each of the individual policy needs to ensure their output lie in the same space so that we can compose. The original absolute coordinate frame with action bounds from dataset statistics might not have satisfy this constraint.. We use the standard Euler angle as the rotation representation. For the joint training baselines, we map the RGB inputs and point cloud inputs to the same feature space with dimension 512 and use a policy head to regress the actions with 1-dimensional UNet [16]. The experiments use point clouds as inputs for inference time, as the RGB inputs behave quite poorly. We train all policies in simulation and the real world with 80000 steps and learning rate 0.0001 with batch size 512 and 64 respectively for around 16 hours on NVIDIA V-100.

For real-world composition, we use a fixed value of $\gamma = 0.03$. We use a T5 encoder [87] to encode the language features and then map to dimension 256 with a single linear layer. The drop ratio and conditional scale for classifier-free training in task composition are 0.1 and 0.01 respectively. We significantly reduce the size of the network by shrinking the embedding dimensions. We use DDPM scheduler with 100 diffusion steps in training time and DDIM scheduler with 16 steps in test time. We use the "squaredcos_cap_v2" schedule with 'beta_end=0.02'. For tuning the composition hyperparameter, we usually start tuning by running the model with a fixed lambda weight, for instance for the classifier free inference, and then we tuned it up/down until the policies became unstable. We also monitored the numeric scale (e.g. mean absolute value of the gradient to the action trajectory) during the composition process.

C. Additional Experiments

In this section, we compare with a more standard feed-forward policy parametrization. In Figure 13 (a), we show that diffusion models can outperform MLP-based policy when testing on both in-domain tools and generalization to new tools. In Figure 13 (b), we also showed additional comparisons on Meta-world [88], a widely used multitask benchmark. In this case, we follow the setup in [27] with 1000 data points for each task in MT-10 (10 tasks) with the same diffusion and policy setup as in the main Fleet-Tools experiments. We use states as policy inputs. Importantly we use the one-hot task vector in the multitask policy training. Figure 12 shows the different scene generalizations in the conducted experiments.

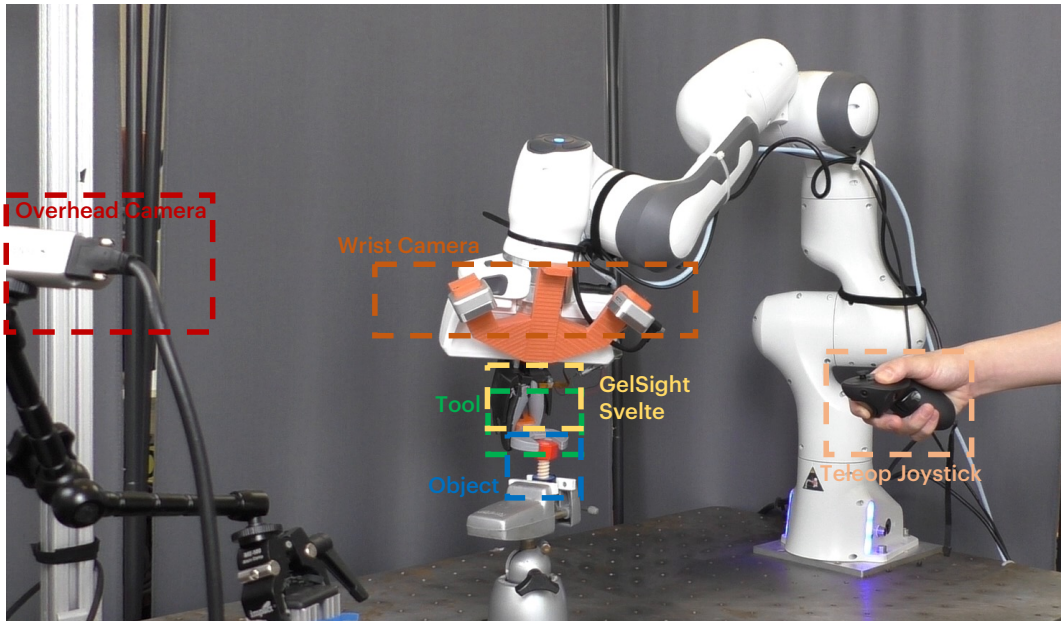

Fig. 11: **Real World Setup**. The robot observation are captured by a wrist camera and an overhead camera. There is a GelSight svelte hand [74] mounted to the end effector grasping a tool to do certain tasks. For the teleoperation, we use an Oculus Quest Pro for tracking the joystick pose and map to the end effector velocity control.

D. Derivation and Discussion

In this section, we give a simple derivation for how the product rule of probability can be used to show Eq. 4, assuming (1) mutual independence of task $\mathcal{T}$ and cost c and (2) conditional independence given the trajectory. Assume that policies trained with different modalities or under different domains, or for different constraints and tasks are independent. The tasks and costs are often conditionally independent given a demonstration trajectory. Taking two separate independent sets of cost functions and tasks $A = (c_1, T_1), B = (c_2, T_2)$ for example, we want to show $p(\tau|A)p(\tau|B) = p(\tau|A, B)p(\tau)$, which shows that $p(\tau|A)p(\tau|B) \propto p(\tau|A, B)$ assuming $p(\tau)$ is uniform. To see this, note that

$$
\begin{aligned}
p(\tau|A)p(\tau|B) &= \frac{p(\tau, A)}{p(A)} \frac{p(\tau, B)}{p(B)} \\
&= \frac{p(A|\tau)p(\tau)}{p(A)} \frac{p(B|\tau)p(\tau)}{p(B)} \\
&= \frac{p(A, B|\tau)p(\tau)}{p(A)} \frac{p(\tau)}{p(B)} \\
&= \frac{p(\tau, A, B)p(\tau)}{p(A, B)} \\
&= p(\tau|A, B)p(\tau) \\
&\propto p(\tau|A, B)
\end{aligned}
$$

Ideally, each cost function and task do not conflict with each other. In the case that these objectives are not independent (or orthogonal under certain structures), but where there are samples that are high likelihood under both objectives, optimizing in the product of the energy landscape will still find such plausible samples (as these samples will have the highest likelihood) in practice.

E. Dataset Collection and Robot Setup

1) Human Dataset

Figure 5 shows our pipeline to process human demonstration data into point cloud action trajectories. We use standard off-the-shelf libraries such as segment-anything [5] and X-Mem [80] for image segmentation and video tracking. We build a simple customized interface to label points on the image in batch and then use the point prompt in segment-anything [5] to extract a single-frame mask. The Segment Anything Model (SAM) pipeline offers various forms of prompts, one of which is a point prompt, which allows us to use a few mouse clicks to initiate a mask prediction for a certain object or tool from the model. After that, the model would output masks in three levels of granularity, and we will pick the middle one as the output mask for this particular instance. The mask is then propagated into XMem [80] for video mask tracking. The mask together with the depth image are used to compute the masked point cloud. The masked point clouds in two frames can be used to compute a relative action using the iterative closest point (ICP in the open3d library). We visualize masked videos and segmented point clouds with action trajectories. Note that no camera calibration is needed and each camera provides an independent data point. The human data collection phase can collect up to 200 trajectories for 20 minutes. One limitation is the quality of the depth point cloud and the distribution shifts from the overhead camera that records the human demonstrations and the wrist camera that is used in robot evaluation time. One way to resolve this is to use camera mounts on human wrists when collecting the data and computing the relative poses via SLAM.

2) Real Robot Dataset

The robot setup has two active cameras (one overhead Intel D-435 and one wrist camera D-405). The overhead camera

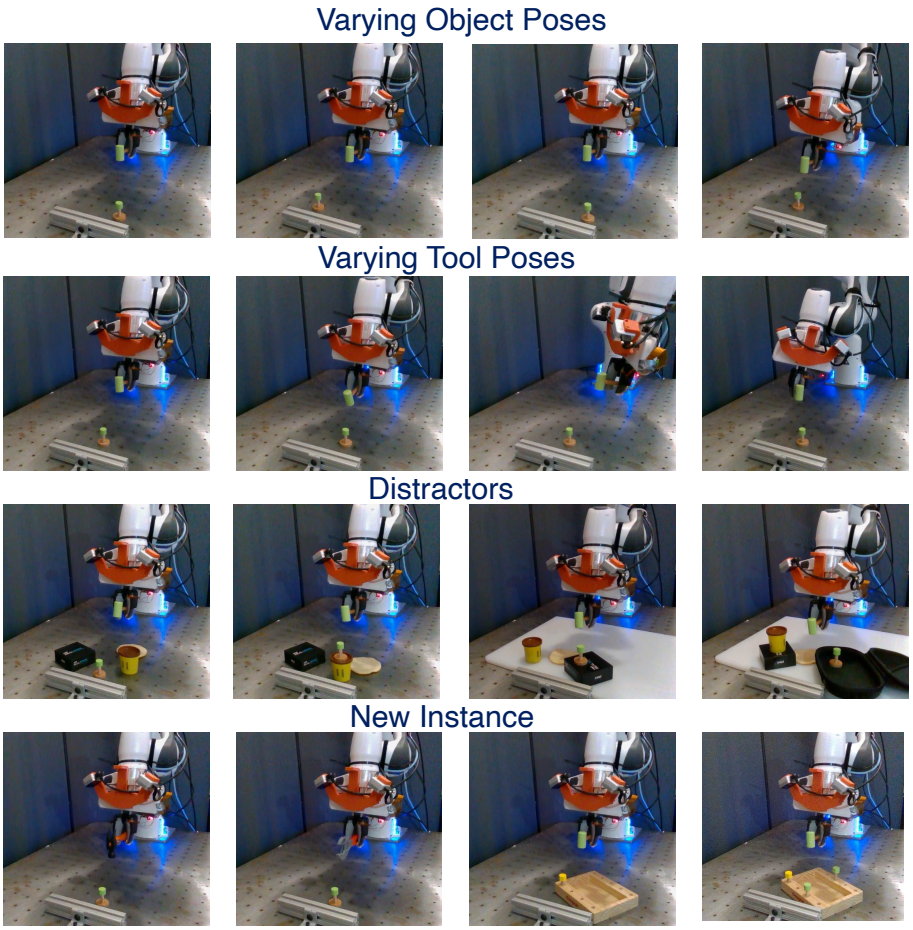

Fig. 12: A list of scene variations considered in the scene-level generalization experiments.

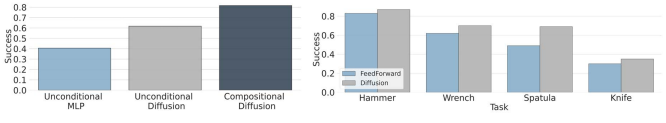

Fig. 13: **(a) Policy Architecture Ablation.** Diffusion outperforms MLP policies in single-task settings in simulation. **(b) Ablation Study on Meta-world.** We conduct additional multitask policy training experiments on the Meta-world benchmark [88]. We observed that classifier-free training can improve the performance in our setups.

is put in front of the robot to have a global view of the robot and the scene and to avoid single-view ambiguity in the policy execution process. The wrist camera view is useful to provide more generalization [1] for the policies as shown in previous works. The robot uses a GelSight Svelte [74] fingers to hold a tool (wrench in this case) to activate an object (nut on the bolt). For the Gelsight Svelte, we note that it's a 3-finger hand that provides a firm grasp of many tool types, and also tactile information is useful for many aspects including force/tactile feedback in the tasks, locating tool-in-hand poses, and robust to environmental changes such as lighting. The RGB policy frequency is 10 and the point cloud perception pipeline frequency is around 5. The end effector and joint position state frequencies are 1000Hz. The tactile image is 30Hz. The tool and object poses are slightly varied for each episode. The real-world point clouds are sampled using farthest point sampling and are accumulated across 5 frames between every reset. After the robot commands the end effector desired pose, the low-level controller sends target joint angles asynchronously and generates a joint-space position trajectory. The end-effector frame action space and the joint velocity are limited to 3 cm per step. We collect between 50 to 100 trajectory demonstrations for each task.

For real-world data collection (Figure 11), we use Oculus Quest Pro to teleop the robot to collect demonstrations. We use standard world-frame velocity control and use extrinsic rotations. The teleop interface also has a switch for re-adjusting the demonstrator's pose and tracking frame reset and finger control functions. We use [89] to solve for Inverse Kinematics of the tracking frame and compute joint-space trajectories asynchronously. The robot controller takes into account the table heights when solving for the inverse kinematics. The

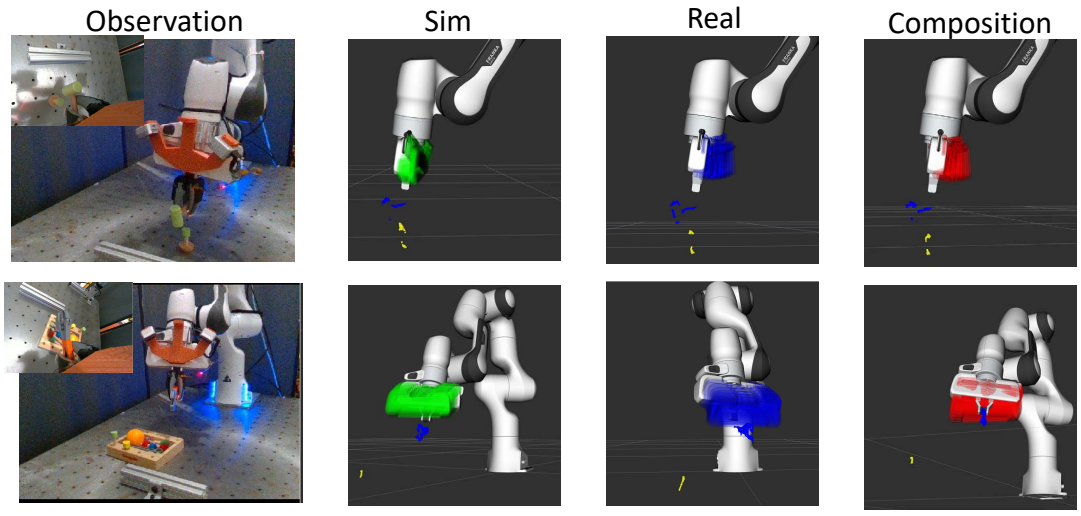

Fig. 14: **Trajectory Composition Snapshots.** The left frame shows the overhead view and the wrist view and the rest three frames show different trajectories predicted. The top row shows that when the objects are very close by, which causes partial observations in depth sensing and segmentation, the simulation policy itself can generate the wrong action trajectories (going forward instead of backward). On the other hand, when the scene drastically changes to a cluttered scene and the tool is changed, the RGB policy can also generate wrong motions. In both cases, the policy composition improves the performance of each constituent policy.

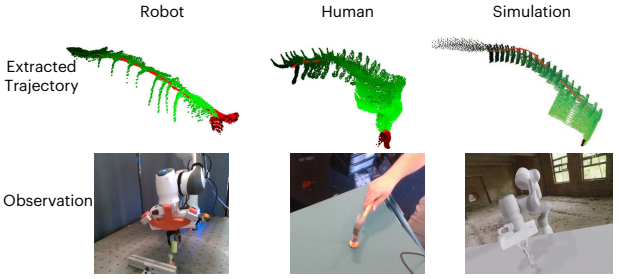

Fig. 15: Comparison of extracted point cloud trajectory (with actions) across different domains. We note that the human data requires no calibration and can be conducted in the wild. The datasets can be collected and processed efficiently.

initialization for each trajectory by default only differs by human resetting errors, while for each generalization axis, we control the experiment variable and change corresponding factors, such as adding a distractor or moving the tool.

3) Simulation Dataset

The simulation datasets follow Fleet-Tools [27] where the expert demonstrations are generated with keypoint trajectory optimizations. More specifically, we label 3 keypoints on the tool and 1 keypoint on the object and then use joint-space KPAM [27] to find a joint-space trajectory for solving each of the tool-use tasks. In total, we collected around 50,000 simulated data points for each of the 5 tool-object pairs across 4 tool families. Each trajectory has frequencies around 6Hz and a length around 25. We apply domain randomization to object poses, tool-in-hand poses, and initial joints in each scene and save fixed scenes for testing. The object models are found in ShapeNet and Objectverse. The language annotations are labeled by humans and the datasets are combined together for multitask training.

F. More Discussions on Heterogeneity

Robots and embodied agents inherently have to deal with heterogeneous inputs and outputs due to the nature of the perception-action loops across diverse environments. We discuss the heterogeneity of data domains in robotics. There are many reasons for this heterogeneity, one of which is different hardware naturally generates slightly different data. In particular, there are three main sources of data domains: simulation, real-robot, and human data. We discuss as follows:

1) **Simulation Data.** Simulation data [11] has the promise of providing huge amounts of diverse data and yet suffers from sim-to-real gaps and the effort and challenges of building complex simulation tasks such as deformable objects. Moreover, simulation provides a useful environment for evaluation and for measuring progress.
2) **Human Demonstration Videos.** Real-world human demonstrations, such as [86], are the biggest existing data source available online for training robots (YouTube for instance) and are easy to acquire. This is important to achieve some surprising results from the success of computer vision and natural language processing. Yet, it suffers from embodiment gaps, especially on contact-rich motions.
3) **Real-world Robot Data.** On-robot data, such as [3], has the least domain shifts, yet it can be prohibitive to acquire at large scales. Even with on-robot data, there is a difference between other robots that share training data and the robot that does the evaluation. For instance, [3] has 22 embodiments and there are different hardware, sensors, and environments associated with each dataset.

Next, we provide a short discussion on some of the heterogeneity in data modalities for robots that interact with the physical world.

1) **Image/Video Data.** RGB Image data is ubiquitous and has the redundancy necessary for representation learning to extract useful structures. It also has the regular data formats and engineering stacks to work with. In the context of robotics, the main issue is that policies learned

from RGB data only can be brittle to irrelevant details such as lighting and backgrounds.

2) **Depth/Point Cloud Data.** Depth and the associated point cloud are lightweight to work with. It allows a simple fusion of history and multiple views and simpler sim-to-real transfer. The problem is that the point cloud will require segmentation and the depth camera can have issues with thin and transparent objects.
3) **Tactile/Haptic Data.** Haptic and tactile data provide useful intrinsic information about the contacts and the fine details. Human uses this information all the time during manipulation. Hardware durability and reproducibility are common issues.
4) **State Data.** State data can contain robot proprioceptive information such as end effector poses, joint angles, velocities, and external torques, which are useful information that summarizes the state of the robots. It can also contain estimated object poses and class information as an abstraction of the world. The problem is that the perception errors will propagate to robot policies.
5) **Language/Goal Data.** Language and other abstracted goal data (such as goal frame) provide useful intent for each task that the robot is commanded to achieve. Yet, it usually misses the low-level details of how to achieve these goals.

Due to the absence of an internet-equivalent and homogeneous dataset for language models in robotics, we believe that the data heterogeneity needs to be considered carefully to use all of such data. Taking the common data modalities RGB and depth, for example, a naive data pooling approach would require paired data. Given two separate depth and color datasets, it's likely to bottleneck the overall dataset size by the dataset with the smaller data amounts. In contrast, our method aims to combine the best of all worlds. There are additional challenges such as data dependence within an episode, data distribution shifts, and long-tail problems, as well as task-level differences. Policies trained with different domains and modalities also can perform differently due to their training distributions. For example, RGB policies can generalize well for a single dexterous task in a fixed scene given enough data but can fail to generalize across details such as lighting. Depth / point-cloud policies trained in simulation can generalize across partial views and randomized scenes but can be challenging in corner cases and contact-rich scenarios. Empirically, the simulation-trained policies also behave less smoothly due to the domain gaps. Tactile-based policies can be useful in contact-rich tasks but require specific data collection procedures.

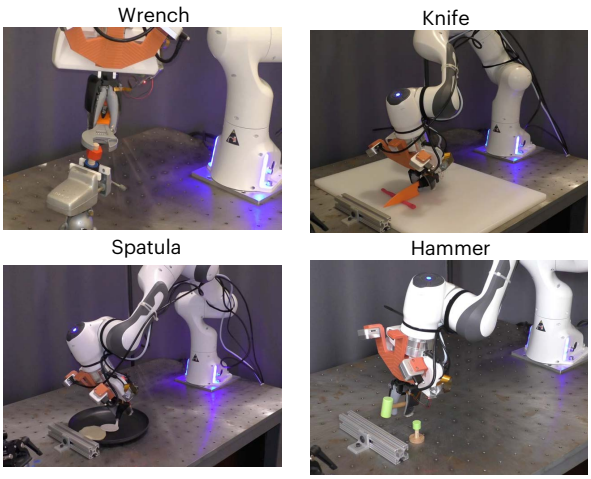

Fig. 16: **Failure Cases**. We have seen some failure cases in fine-grained tasks such as the wrench task and bad contact interaction such as in the spatula and knife tasks.