JUICER: Data-Efficient Imitation Learning for Robotic Assembly

Lars Ankile[1,3], Anthony Simeonov[2,3], Idan Shenfeld[2,3], Pulkit Agrawal[2,3]
[1]Harvard University, [2]Massachusetts Institute of Technology, [3]Improbable AI Lab

***Abstract*— While learning from demonstrations is powerful for acquiring visuomotor policies, high-performance imitation without large demonstration datasets remains challenging for tasks requiring precise, long-horizon manipulation. This paper proposes a pipeline for improving imitation learning performance with a small human demonstration budget. We apply our approach to assembly tasks that require precisely grasping, reorienting, and inserting multiple parts over long horizons and multiple task phases. Our pipeline combines expressive policy architectures and various techniques for dataset expansion and simulation-based data augmentation. These help expand dataset support and supervise the model with locally corrective actions near bottleneck regions requiring high precision. We demonstrate our pipeline on four furniture assembly tasks in simulation, enabling a manipulator to assemble up to five parts over nearly 2500 time steps directly from RGB images, outperforming imitation and data augmentation baselines.**

I. Introduction

Automatic assembly is one of the most practically valuable applications of robotic technology. However, robots have demonstrated limited utility in production environments where a wide variety of products are manufactured in small quantities (i.e., "high-mix, low-volume"). Robot learning has the potential to transform this landscape. For instance, a programmer can directly "teach" new assembly sequences via demonstration rather than scripting exact trajectories. Learned controllers that work directly with sensor data can also reduce reliance on custom part-positioning fixtures. However, learning to perform assembly from raw perception has remained challenging [1], as assembly requires long-horizon manipulation with high precision requirements.

To train closed-loop assembly policies that use RGB images, one could try using Reinforcement Learning (RL), but RL struggles with long task horizons and sparse rewards. On the other hand, while learning from demonstrations is more tractable, requiring many demonstrations creates a significant data collection burden. Instead, we consider Imitation Learning (IL) in a small-data regime that enables users to collect demonstration data themselves ($\sim$50 demonstrations). One could also combine IL with RL [2] or fine-tune a pre-trained multitask model [3], but those investigations introduce additional complexity and are beyond the scope of this work. Thus, the primary challenge is to extract as much information and performance as possible from a modestly sized, manually collected dataset of assembly trajectories. Our goal is to understand the resulting challenges that emerge and suggest a pipeline that improves Imitation Learning when operating with a modest number of demonstrations.

For one, effectively fitting a complex set of demonstrated actions while operating from raw images can be challenging for long-horizon tasks requiring high precision. The choice of policy architecture and action prediction mechanism has a tremendous impact on the ability to fit the data well. Building off recent work, we provide additional evidence for the advantages of representing policies as conditional diffusion models [4]–[7] and predicting chunks of multiple future actions [6], [8].

Another difficulty is learning robust behaviors around "bottleneck" regions where even slight imprecisions can lead to failure. For example, mistakes during part insertion frequently lead to dropped parts and overall task failure. One strategy to mitigate this is to expand dataset support via structured data augmentation and noising. The main idea is to supervise the model with locally corrective actions that return to the training distribution from synthetically perturbed states [9]–[12]. However, such methods often make assumptions that prevent their direct application to visuomotor policy learning for multi-step assembly (e.g., known object poses, static scenes, assuming no object in hand). We deploy a similar strategy that requires a weaker assumption - the ability to reset the system to a given state in the demonstrations. Our current work uses a simulator to perform such resets (and also evaluates learned policies in simulation), but automatic resets can be deployed in real-world settings as well [13]. We reset the scene to bottleneck states, perturb the scene by simulating random "disassembly" actions, and synthesize corrective actions while rendering new scene images. The corrective actions are obtained by reversing the disassembly sequence [14]–[17]. This enables structured data noising in a broader class of scenarios.

Finally, we consider opportunities to automatically expand the dataset of whole trajectories. We take advantage of the iterative model development cycle across tasks to achieve this. Namely, successful or partially successful rollouts are often collected during model evaluation if the policy has a non-zero success rate. New data may also be collected (either via demos or evaluation rollouts) for other tasks introduced in parallel. We use these in an IL-based "collect and infer" setup [18]–[20] and obtain larger, more diverse datasets without incurring additional human effort.

In summary, we propose JUICER, a pipeline for learning high-precision manipulation from a small number of demonstrations. Our pipeline combines the benefits of diffusion policy architectures and mechanisms for automatically expanding dataset size via data noising and iterative model development cycles. We show the technique's effectiveness on four simulated tasks from the FurnitureBench benchmark [1] for long-horizon furniture assembly, including tasks with

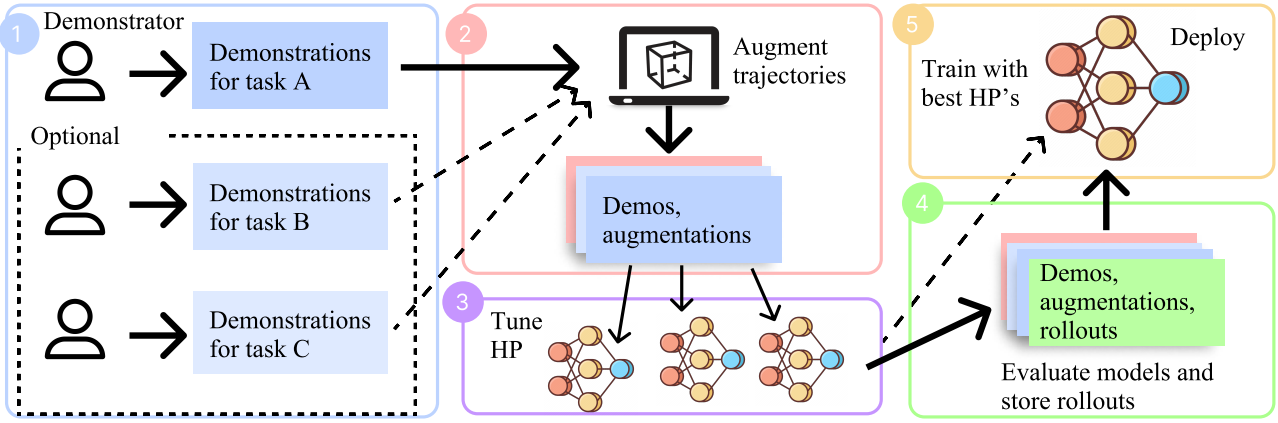

Fig. 1: Overview of the proposed approach. (1) Collect a small number of demonstrations for the task (and related tasks, if available). (2) Using annotations of *bottleneck states*, augment the demonstration trajectories to create synthetic corrective actions and increase coverage around the bottleneck states. (3) Use this dataset to train models with different hyperparameters. (4) Store all rollouts throughout model evaluations. (5) Add any successful rollout to the training set and train the best architecture on all data amassed.

horizons up to ∼2500 timesteps and assemblies of up to 5 parts to be precisely grasped, oriented, and inserted. Our results show that each component of our system improves overall task success compared to vanilla IL baselines. Our ablations illustrate how different design choices impact overall system performance, and we demonstrate that JUICER enables learning high-performance policies using as few as 10 human demonstrations. Finally, we contribute our bottleneck state labeling tool, trajectory augmentation tool, collected demonstration datasets, and IL-based collect-and-infer pipeline for the community to utilize and build on. Supplementary videos and Appendix can be found at https://imitation-juicer.github.io/.

II. Preliminaries

A. Notation and System Components

We aim to learn a policy $\pi_\theta(a_t|o_t)$ mapping observations o_t to actions a_t using a dataset of trajectories, $\mathcal{D} = \{\tau_1, ..., \tau_N\}$, $\tau_i = \{(o_1, a_1), ..., (o_T, a_T)\}$, with T being the trajectory length. Our observations include RGB images (front and wrist views) and the robot's proprioceptive state. Our actions represent desired 6-DoF end-effector poses, reached via a low-level Operation Space Controller [21]. We train π_θ with Behavior Cloning (BC), i.e., we optimize θ to maximize the likelihood of the data, $\underset{\theta}{\text{argmax}}\ \mathbb{E}_{(a_t,o_t)\sim\mathcal{D}}\left[\log \pi_\theta(a_t|o_t)\right]$.

The policy input and output can consist of consecutive observation or action chunks. Denoting T_o as the number of observations we condition on (i.e., up to and including the current step), we pass $\mathbf{o_t} = [o_{t-T_o}, ..., o_t]$ to π_θ, with each o_t of dimension $|o|$. Let T_p be the number of predicted future actions, i.e., the policy outputs a sequence of actions $\mathbf{a_t} = [a_t, ..., a_{t+T_p}]$, with each a_t of dimension $|a|$. Although predicting an action chunk $\mathbf{a_t}$ of length T_p, we only execute a subset $[a_t, ..., a_{T_a}]$, with execution horizon $T_a \leq T_p$.

B. Denoising Diffusion Probabilistic Models

Our method uses Denoising Diffusion Probabilistic Models (DDPM) [22]–[24] to learn high-dimensional distributions. The data-generating process is modeled with a fixed "forward" noising process $q(x_{k+1}|x_k) = \mathcal{N}(x_{k+1}; \sqrt{\alpha_k}x_k, (1-\alpha_k)\mathbf{I})$ and a learned reverse process $p_\theta(x_{k-1}|x_k) = \mathcal{N}(x_{k-1}|\mu_\theta(x_k, k), \Sigma_k)$. A sample is denoted x_0, and $x_1, ..., x_{k-1}$ are latent variables in the generation process ending as pure Gaussian noise $x_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with an appropriate choice of noise schedule α_k. Using $\bar{\alpha}_k = \prod_t^k \alpha_s$, we can sample any x_k step k as $q(x_k|x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_k}x_0, (1-\bar{\alpha}_k)\mathbf{I})$, allowing the reverse process to be estimated through a "noise prediction" surrogate loss [22],

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{k\sim[K] \\ x_0\sim q \\ \epsilon\sim\mathcal{N}(0,I)}} \left[\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_k}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_k}\epsilon, k) \|^2\right].$$

The predicted noise ϵ_θ is represented with a neural network. After training has converged with the above objective, we can sample from the distribution through an iterative "refinement" process, starting from Gaussian noise x_K, and produce x_0 by subtracting the predicted noise repeatedly:

$$x_{k-1} = \frac{1}{\sqrt{a_k}}\left(x_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(x_k, k)\right) + \sigma_k\mathbf{z}$$

The DDIM sampler [25], [26], allows a model trained with the above DDPM objective to be sampled deterministically with fewer steps.

III. Methods

A. System Overview

We propose JUICER, a data-efficient Imitation Learning (IL) pipeline for precise and long-horizon assembly from image observations in the simulator, presented in Figure 1. The pipeline starts by **(1)** collecting a small number of task demonstrations (Section III-B) (as well as any available related tasks). The demonstrator also annotates states that require extra precision. These labels are used to **(2)** synthetically increase the data support around these states (Section III-C). To model diverse human-collected and synthetic data, we **(3)** fit a Diffusion Policy (Section III-D). As models are evaluated, any (partially) successful trajectory is **(4)** stored and added back to the training set (Section III-E),

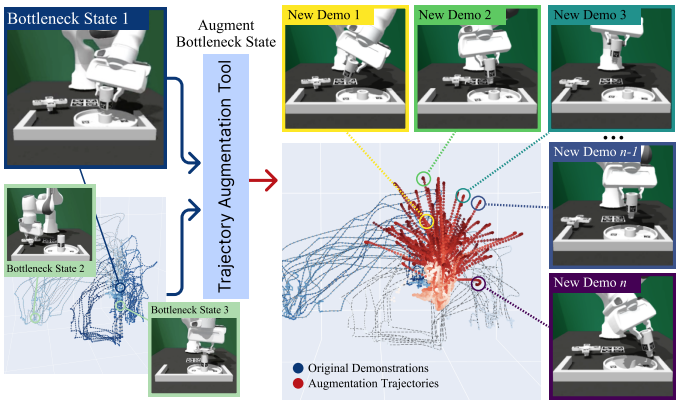

Fig. 2: An example of extracting *"bottleneck"* states and using trajectory augmentation tool to create an arbitrary amount of counterfactual trajectories near the bottleneck, effectively increasing the data support and teaching corrective actions.

further increasing support. Using the best hyperparameters and all data amassed in the process, we **(5)** train a final model for initial deployment.

B. Data Collection and Annotation

The first step is robot teleoperation. While any teleoperation interface can be used, our system utilizes a 3DConnexion SpaceMouse for 6-DoF end-effector control. For each task, 50 demos are collected, taking 1-3 hours per task. Teleoperation in simulation offers unique opportunities to make demo collection more seamless and improve user experience. For instance, using the ability to perform resets, we implement an "undo" button and enable users to correct their mistakes while collecting trajectories. These additions save time by reducing the number of trials discarded due to operator failure.

After data collection, we add sparse annotations of "bottleneck" state transitions requiring high precision (these labels are used for data augmentation, see Section III-C) To annotate the demonstrations, a user loads the successful trajectories, steps through each episode, and marks the timesteps when bottleneck states occur. This step adds ∼15% to the total data collection time, and even the longer demos (∼2500 time steps) can be fully annotated in ∼1-2 minutes.

C. Trajectory Augmentation

Our experiments showed that small misalignment errors during precise grasps or insertions could quickly compound and lead to catastrophic task failure (e.g., parts falling out of reach in extreme out-of-distribution poses). We, therefore, want to improve the model's ability to reach and transition through these bottleneck regions robustly. However, we want to avoid having the demonstrator manually provide examples of additional recovery behaviors [27], [28]. Our solution is to increase dataset support in regions that require high precision via synthetic data augmentation (see Algorithm 1). The first step is to reset the simulated robot and environment to a recorded bottleneck state from a demonstration. We then sample a random end-effector pose *away* from the bottleneck state and construct a sequence of "disassembly"

Algorithm 1: Backward Trajectory Augmentation

Input: Spherical coordinate limits, $\mathbf{r} = \{r_{\min}, r_{\max}\}$, $\boldsymbol{\theta} = \{\theta_{\min}, \theta_{\max}\}$, $\boldsymbol{\phi} = \{\phi_{\min}, \phi_{\max}\}$;
Dataset of human-collected demos $\mathcal{D}_H$
Result: Augmented trajectories $\mathcal{D}_A = \{\tau_i\}_{i \in [N_A]}$

Init: $i \leftarrow 1$, $\mathcal{D}_A = \varnothing$
while $i < N_A$ **to** N_p **do**
  Sample demonstration trajectory $\tau_H \sim \mathcal{D}_H$
  Sample bottleneck state $s_{\text{bottleneck}} \sim \tau_H$
  Reset world state to $s_{\text{bottleneck}}$
  Sample a state s_{target} within the limits $\mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\phi}$
  Command EE from $s_{\text{bottleneck}}$ to s_{target} and record inverse actions τ_i^{cand}
  Execute τ_i^{cand} from s_{target}, ending in state s'
  if $\| s' - s_{bottleneck} \|^2 \leq \epsilon$ **then**
    Store trajectory $\mathcal{D}_A \leftarrow \mathcal{D}_A \cup \{\tau_i^{\text{cand}}\}$
    $i \leftarrow i + 1$
  end
  else
    Discard candidate trajectory τ_i^{cand}
  end
end
return $\mathcal{D}_A$

actions that reach this pose via linear interpolation/SLERP. Finally, starting from the random pose, the robot executes a "correction" that returns from the randomly perturbed state to the original bottleneck. This correction is obtained by *reversing* the disassembly actions. We record the corrective actions and rendered scene images and compare the resulting scene state to the original demonstration. If the final states match, the trajectory segment is saved in the same format as the demonstrations to be used for BC training.

Figure 2 shows an example of this procedure for the `round_table` task. Note how the augmented trajectory dataset provides a "funnel" toward the critical state required for successful insertion. This blends prior techniques for guiding tasks like assembly [17], insertion [29], [30], and kitting [16], together with structured data noising that has been deployed to reduce covariate shift in IL [9], [12].

D. Policy Design Choices

1) Data Processing and Image Encoding: We normalize all actions and proprioceptive states so each dimension lies in the range $[-1, 1]$ using $\min$ and $\max$ statistics from demo data across all tasks. Consistent with prior work, we have found that position control greatly outperforms delta control [6]. We represent orientations using the 6-dimensional representation [31], [32] resulting in an action space of $|a_t| = 10$. When processing image observations, we first resize raw RGB images from 1280×720 to 320×240. We then perform various random crops and color jitter to increase dataset diversity and reduce overfitting, and perform a final resizing down to 224×224.

We encode the processed images with a vision model. The encoder outputs latent image representations that we project

to 128-dimensional vectors (one for each camera view). The final observation provided to the policy is of size $|o_t| = 2 \cdot 128 + 16 = 272$. Contrary to [6], we find an ImageNet-pre-trained `ResNet18` with global pooling to outperform a `ResNet18` trained from scratch with `SpatialSoftmax` pooling, similar to [7]. Each `ResNet18` has ∼11 million parameters.

2) Diffusion Policy: Given a dataset of observation-action chunks, $\mathcal{D} = \{(\mathbf{o}_i, \mathbf{a}_i)\}_{i \in [N]}$, the goal is to fit a policy model $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$. The standard denoising objective in fitting diffusion models from [22] is adapted to the setting of modeling an action distribution conditioned on an observation, $p(\mathbf{a}_t|\mathbf{o}_t)$, but where the noise model is parameterized as $\epsilon_\theta(\mathbf{a}_t, \mathbf{o}_t, k)$. Both the action and observation are chunks along the time-dimension, $\mathbf{a}_t \in \mathbb{R}^{T_p \times |a_t|}$ and $\mathbf{o}_t \in \mathbb{R}^{T_o \times |o_t|}$. We find longer prediction horizons benefit performance and use $T_p = 32$ in all our experiments. We still find the action execution horizon of $T_a = 8$ provides a good tradeoff between open-loop consistency and closed-loop reactivity. Consistent with recent work [20], we have not found including history helpful and use $T_o = 1$ for all tasks.

Like prior work [6], [33], we use a 1D temporal CNN-based U-Net introduced by [4], with the same FiLM conditioning layers in the residual blocks used in [6]. Experiments with the time-series diffusion transformer introduced in [6] did not improve performance in our tasks. Unless stated otherwise, our experiments use a U-Net with 3 downsampling blocks of dimension 256, 512, and 1024, yielding a model with 69 million parameters and a full policy model with 91 million parameters. At training time, we used 100 denoising steps for the DDPM sampler [22] and 8 denoising steps with the DDIM sampler [25], [26] at inference time.

E. Dataset Expansion with "Collect-and-Infer"

After training on the combined teleop and augmentation trajectories, we can deploy initial versions of the assembly controller and evaluate their performance. Through this process, we expand the dataset by re-incorporating any successful rollouts into the training set and training a new model on the expanded dataset. Concretely, starting with the initial dataset $\mathcal{D}_H$ (of 50 demos), we fit k policies $\pi^i_{\mathcal{D}_H}, i \in [k]$ with different hyperparameters. We then evaluate these policies and store any successful trajectories in $\mathcal{D}_E = \{\tau^j_i\}, i \in [k], j \in [n_{\text{eval}}]$. Finally, we take the best policy and train a new one from scratch with the same hyperparameters on all the data, $\mathcal{D}_{H+E} = \mathcal{D}_H \cup \mathcal{D}_E$. This paper focuses on the initial effects of the technique and limits the rollouts used to 50 (experiments with more than 50 rollouts are included in the Appendix).

This is reminiscent of the "collect-and-infer" paradigm for RL [34], [35] and IL [20] which has shown to be a simple but powerful mechanism for iterative policy improvement. We find that collect-and-infer complements the augmentation procedure discussed Section III-C. Namely, trajectory augmentation is directly applied to original demonstrations and does not require collecting new rollouts. We find, though, that performance improvements saturate unless more full trajectory data is available. On the other hand, while collect-and-infer requires an initial policy with non-zero success, it provides valuable extra supervision of *complete* trajectories. Our experimental results show that combining both dataset expansion methods boosts overall performance the most.

F. Multitask Learning

Prior work has shown positive transfer between different tasks on both large [36] and more modest scales [20]. Similarly, our final step involves co-training on data from other related assembly tasks. This is inspired by observations in other prior works that show that multitask training benefits performance even at a modest dataset scale (100-1000 demos) [20], [37].

IV. Experiments and Results

A. Experimental Setup

1) Tasks: We choose 4 tasks from the benchmark FurnitureBench [1]: `round_table`, `lamp`, `square_table`, and `one_leg` (where `one_leg` is a subtask of the `square_table`, see Figure 3). This task subset presents a variety of challenges which are summarized in Table I.

TABLE I: Task Attribute Overview

	Round Table	Lamp	Square Table	One Leg
Episode Length	1,100	900	2,500	550
# Parts to Assemble	3	3	5	2
Dynamic Object	✗	✓	✗	✗
# Precise Insertions	2	1	4	1
Precise Grasping	✓	✗	✗	✗
Insertion Occlusion	✓	✗	✗	✗

2) Training:

a) Default Training protocol: For each task and dataset, we train 5 models (as described in Section III-D) from different seeds for up to 160,000 gradient steps with the `AdamW` optimizer [38]. We use a learning rate of 10^{-4} and a cosine learning rate schedule [39] with 500 warmup steps and a batch size of 256. To avoid overfitting, we terminate if validation loss does not improve for 5 epochs.

During training, we apply image augmentations to both the wrist and front camera images, similar to [40], [41]. We apply the PyTorch color jitter augmentation with the parameters for brightness, contrast, saturation, and hue, all equal to 0.3, and Gaussian blur with kernel size 5 and $\sigma \in [0.01, 2]$ to both views. We also apply random image cropping, but only on the front view. We do not crop the wrist view because the end-effector action labels are not invariant to randomly translated crops of the wrist view.

b) Multitask Training: To investigate the impact of multitask co-training, we train models on mixes of 50 demonstrations for each of the `round_table`, `square_table`, and `lamp` tasks, for a total of 150 demos.

3) Evaluation: We evaluate the 5 models trained from different random initializations per condition on 100 rollouts and report the mean and max *success rate*, i.e., the share of attempts that result in a complete assembly. We also examine what performance can be achieved using an even smaller budget of human-collected demos (see Section IV-B.7).

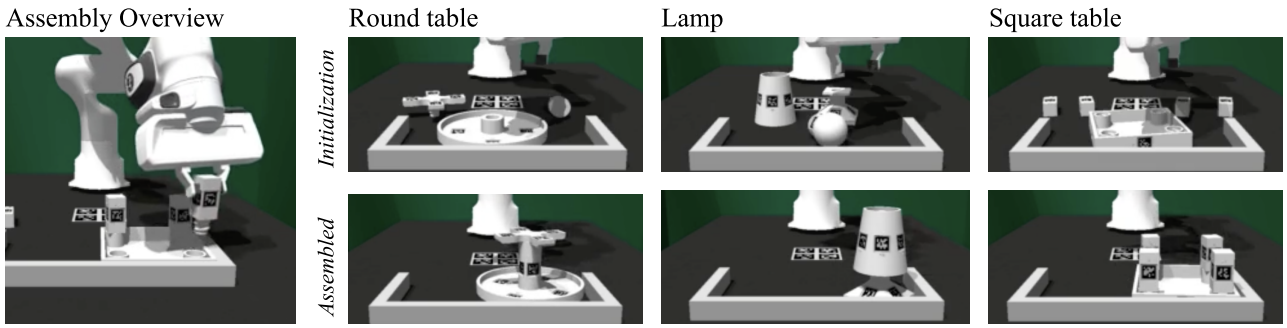

Fig. 3: Overview of the tasks. The first row shows a random initialization of the parts, and the second row shows the full assembly.

4) Baselines: The baseline architecture is a residual MLP model with 5 residual blocks (∼10 million parameters). It receives the same features as the diffusion model, i.e., image encoding vectors and the proprioceptive state. The first baseline is an MLP without action-chunking, i.e., $T_p = T_a = 1$. The second uses the same action-chunking as the diffusion policy, $T_p = 32$ and $T_a = 8$. We also compare against the noising approach proposed in [9], which suggests expressing robot states in an object-centric coordinate frame and perturbing robot states to synthesize corrective action labels and reduce covariate shift. Under our setting, where the policy cannot access object poses, this reduces to injecting noise *only* in the robot's proprioceptive state and leaving the images unmodified.

B. Numerical Results: Success rates

Figure 4 shows success rates across tasks and methods.

1) MLP Baseline and Diffusion Policy: The MLP policy without action chunking (MLP-NC) could not complete any tasks. With action-chunking (MLP-C), however, the MLP policy drastically improves, achieving average success rates of 40%, 9%, 3%, and 2% for `one_leg`, `round_table`, `lamp`, and `square_table`, respectively. The performance difference between predicting the next action $T_p = 1$ versus predicting a chunk $T_p = 32$ further corroborates that inducing long-term action consistency is highly beneficial for IL [6], [8]. We also observe the diffusion policy (DP-BC) to improve performance beyond the MLP. The relative performance improvement is greater for the tasks with longer horizons, possibly due to a stronger ability to stay in-distribution [11], somewhat mitigating compounding errors.

2) Proprioceptive State Noise Injection: The results from injecting noise into the robot state [9] are mixed: State noising reduces performance for `round_table`, makes little difference for `lamp`, but improves performance for `one_leg` and `square_table` over the standard BC model. We hypothesize that this reflects the different relative utility of proprioceptive state information depending on the task. For instance, the proprioceptive state is less useful than the egocentric view during insertion in `square_table`, whereas for `round_table`, the proprioceptive state is likely more helpful because the wrist camera image often becomes occluded by the larger grasped part during insertion. It appears that noise injection harms performance when reliable robot states are critical, but in other cases, it likely offers a beneficial form of regularization, i.e., as discussed in [42].

3) Trajectory Augmentation (TA): Adding synthetic data around the bottleneck states to the original dataset of 50 teleoperation demonstrations improves performance, especially for `round_table` and `one_leg`. We believe that TA provides larger performance gains on these tasks because the primary bottleneck in `round_table` and `one_leg` involves precise insertion. In contrast, the main challenge in `lamp` is to grasp a dynamic rolling bulb, and our augmentation procedure is currently less suited to overcoming challenges with dynamic objects. For `square_table`, the task comprises a long sequence of sub-tasks, many of which are not insertions and thus currently not addressed by the augmentation. Thus, only focusing on bottleneck states creates a more delicate balance between increasing support for some phases and not washing out other phases.

4) Collect-and-Infer (CI): We see a large improvement when training on a mix of 50 teleop demos and 50 successful rollouts collected from arbitrary model evaluations. The improvement is most marked for the `lamp` task, where the success rate increases 3×. This is attributable to the dynamic bulb part that rolls on the table and reaches different table regions depending on how the robot and other nearby parts interact with it. By training on successful rollouts, the policy observes a wider distribution of initial bulb positions. We also see a doubling of performance for `square_table`, which is likely attributable to the long-horizon and multi-faceted nature of the task, which requires more uniform coverage of the state space throughout the episode.

5) Traj. Aug. & Collect-and-Infer (TA & CI): We hypothesize that synthetic augmentations and rollout trajectories are mutually beneficial, as more augmentations can be added without washing out other phases. Indeed, the results in **??** show that combining TA & CI leads to the overall best average performance across all tasks, achieving averages of 76%, 32%, 29%, and 15% on `one_leg`, `round_table`, `lamp`, and `square_table`, respectively.

6) Multitask Training: For multitask training, we find that a *single* model trained on a mix of 50 demonstrations from each of `round_table`, `lamp`, and `square_table` outperforms all baseline diffusion policy (DP-BC) models trained individually on each respective task. The ability to improve performance from 18% to 25% for `round_table` and 6% to 14% for `lamp` merely by mixing data from

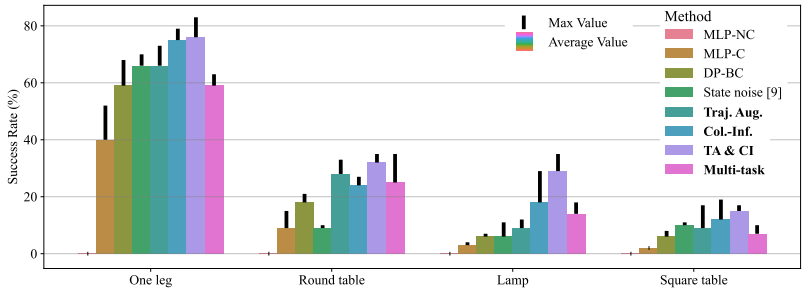

Fig. 4: Average and maximum success rates (%) of methods across tasks. Bolded methods are components of our JUICER pipeline.

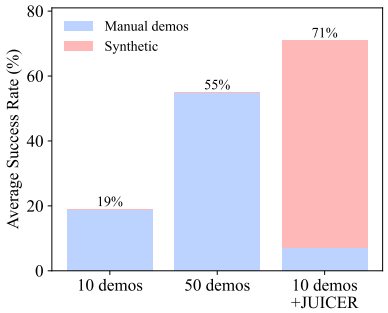

Fig. 5: Learning `one_leg` from 10 human demos.

different tasks suggests that the model implicitly learns certain skills shared between the tasks.

7) Further Minimizing Demonstration Budget: Finally, we perform one full JUICER iteration starting from just 10 demonstrations of the `one_leg` task and compare it with the performance obtained from 50 human-collected demonstrations. Figure 5 shows that training the baseline diffusion policy on only 10 demonstrations for the `one_leg` task yields a success rate of 19%, compared to 59% with 50 demonstrations. However, models trained on a combined set of 10 human demonstrations, 90 successful rollouts, and 100 augmentations achieve, on average, 71% success, a $> 10\%$ improvement beyond collecting 50 demos by hand.

V. Related Work

A. Improving Robustness of Behavior Cloning

There have been many proposed techniques for perturbing states and obtaining corrective actions to expand dataset support. In [9], system states are randomly corrupted, and dataset actions are re-used as corrective actions. [10] generalizes this by learning local dynamics models that support synthesizing recovery actions. Each system relies on a fully observed state space and does not readily apply when operating with image observations and unknown object poses.

The method in [12] trains a Neural Radiance Field to synthesize wrist-camera views that would result from perturbed end-effector states. Perturbations are then used to construct corrective action labels. However, [12] operates with static scenes and assumes an image masking ability that is more challenging for a policy that must insert an unknown grasped object. Other strategies may involve augmented loss terms [43] or requiring experts to directly demonstrate corrective actions [28], [44]. Each of these requires assumptions and complexities that our system does not incur.

B. Diffusion for Decision-Making

Diffusion models [22]–[24] have been shown to provide significant performance improvements in a variety of sequential decision-making and control domains [4], [6], [7], [33], [45]. There are mechanisms for predicting actions at high frequencies with diffusion [6], and such models have supported the development of theoretical performance guarantees for BC [46]. Our results similarly show an expressive diffusion policy architecture benefits performance in the challenging setting of multi-step assembly from images.

C. Behavior Learning for Insertion and Assembly

Much past work has studied the related problems of assembly, insertion, and kitting. InsertionNet [14], [15] reverses random actions taken from a manually demonstrated "inserted" state to supervise robust insertion behaviors. Form2Fit [16] uses self-supervised disassembly sequences to learn pick-and-place affordance models for kitting tasks. Assemble-them-all [17] and ASAP [47] also leverage disassembly to guide the search for a sequence and motion that achieves complex multi-part assembly. Our work generalizes the same principle for data augmentation around bottleneck transitions while operating on continuous low-level actions and training from complete demonstration trajectories.

VI. Limitations and Conclusion

Limitations: While simulation offers our pipeline many benefits, we have not yet adapted our approach to the real world. One potential way forward is to directly transfer our learned policies using domain randomization. We are also interested in real-world augmentation/collect-and-infer strategies, which may require additional components for success classification and resets. We also cannot yet generalize to arbitrary initial part pose distributions. However, there is potential of combining collect-and-infer with a curriculum that gradually increases the initial state distribution.

Conclusion: This paper proposes a mechanism for training a robot to perform precise, multi-step assembly tasks in simulation using a small demonstration data budget. We develop a pipeline combining expressive policy architectures with tools for synthetic dataset expansion and corrective action generation to overcome challenges with limited dataset size, covariate shift, and task complexity. Our results illustrate that the pipeline can use a modest number of demonstrations to acquire policies for constructing multiple complex assemblies that require precise manipulation behaviors executed over long horizons.

Acknowledgment

The computations in this paper were run on the FASRC cluster, supported by the FAS Division of Science Research Computing Group at Harvard University, and on the MIT Supercloud [48].

REFERENCES

[1] M. Heo, Y. Lee, D. L. Kaist, and J. J. Lim, "FurnitureBench: Reproducible Real-World Benchmark for Long-Horizon Complex Manipulation," 2023, arXiv: 2305.12821v1. [Online]. Available: https://clvrai.com/furniture-bench

[2] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," https://octo-models.github.io, 2023.

[4] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with Diffusion for Flexible Behavior Synthesis," Dec. 2022, arXiv:2205.09991 [cs]. [Online]. Available: http://arxiv.org/abs/2205.09991

[5] A. Ajay, S. Han, Y. Du, S. Li, A. Gupta, T. Jaakkola, J. Tenenbaum, L. Kaelbling, A. Srivastava, and P. Agrawal, "Compositional Foundation Models for Hierarchical Planning," Sept. 2023, arXiv:2309.08587 [cs]. [Online]. Available: http://arxiv.org/abs/2309.08587

[6] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," June 2023, arXiv:2303.04137 [cs]. [Online]. Available: http://arxiv.org/abs/2303.04137

[7] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin, "Imitating Human Behaviour with Diffusion Models," Mar. 2023, arXiv:2301.10677 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2301.10677

[8] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[9] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, and S. Srinivasa, "Grasping with Chopsticks: Combating Covariate Shift in Model-free Imitation Learning for Fine Manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 6185–6191. [Online]. Available: https://ieeexplore.ieee.org/document/9561662/

[10] L. Ke, Y. Zhang, A. Deshpande, S. Srinivasa, and A. Gupta, "CCIL: Continuity-based Data Augmentation for Corrective Imitation Learning," Oct. 2023, arXiv:2310.12972 [cs]. [Online]. Available: http://arxiv.org/abs/2310.12972

[11] A. Block, A. Jadbabaie, D. Pfrommer, M. Simchowitz, and R. Tedrake, "Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[12] A. Zhou, M. J. Kim, L. Wang, P. Florence, and C. Finn, "Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 907–17 917.

[13] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6664–6671.

[14] O. Spector and D. D. Castro, "InsertionNet - A Scalable Solution for Insertion," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5509–5516, July 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9420246/

[15] O. Spector, V. Tchuiev, and D. Di Castro, "InsertionNet 2.0: Minimal Contact Multi-Step Insertion Using Multimodal Multiview Sensory Input," in *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022, pp. 6330–6336. [Online]. Available: https://ieeexplore.ieee.org/document/9811798/

[16] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2fit: Learning shape priors for generalizable assembly from disassembly," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9404–9410.

[17] Y. Tian, J. Xu, Y. Li, J. Luo, S. Sueda, H. Li, K. D. Willis, and W. Matusik, "Assemble them all: Physics-based planning for generalizable assembly by disassembly," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–11, 2022.

[18] M. Riedmiller, J. T. Springenberg, R. Hafner, and N. Heess, "Collect & infer-a fresh look at data-efficient reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 1736–1744.

[19] T. Lampe, A. Abdolmaleki, S. Bechtle, S. H. Huang, J. T. Springenberg, M. Bloesch, O. Groth, R. Hafner, T. Hertweck, M. Neunert, *et al.*, "Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots," *arXiv preprint arXiv:2312.11374*, 2023.

[20] K. Bousmalis, G. Vezzani, D. Rao, C. M. Devin, A. X. Lee, M. B. Villalonga, T. Davchev, Y. Zhou, A. Gupta, A. Raju, *et al.*, "Robocat: A self-improving generalist agent for robotic manipulation," *Transactions on Machine Learning Research*, 2023.

[21] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[22] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," Dec. 2020, arXiv:2006.11239 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2006.11239

[23] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[24] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[25] J. Song, C. Meng, and S. Ermon, "Denoising Diffusion Implicit Models," Oct. 2022, arXiv:2010.02502 [cs]. [Online]. Available: http://arxiv.org/abs/2010.02502

[26] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5775–5787, 2022.

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

[28] D. Brandfonbrener, S. Tu, A. Singh, S. Welker, C. Boodoo, N. Matni, and J. Varley, "Visual Backtracking Teleoperation: A Data Collection Protocol for Offline Image-Based Reinforcement Learning," Oct. 2022, arXiv:2210.02343 [cs]. [Online]. Available: http://arxiv.org/abs/2210.02343

[29] O. Spector and D. Di Castro, "Insertionnet-a scalable solution for insertion," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5509–5516, 2021.

[30] O. Spector, V. Tchuiev, and D. Di Castro, "Insertionnet 2.0: Minimal contact multi-step insertion using multimodal multiview sensory input," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6330–6336.

[31] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

[32] J. Levinson, C. Esteves, K. Chen, N. Snavely, A. Kanazawa, A. Rostamizadeh, and A. Makadia, "An analysis of svd for deep rotation estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 554–22 565, 2020.

[33] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is Conditional Generative Modeling all you need for Decision-Making?" July 2023, arXiv:2211.15657 [cs]. [Online]. Available: http://arxiv.org/abs/2211.15657

[34] M. Riedmiller, J. T. Springenberg, R. Hafner, and N. Heess, "Collect & Infer – a fresh look at data-efficient Reinforcement Learning," Aug. 2021, arXiv:2108.10273 [cs]. [Online]. Available: http://arxiv.org/abs/2108.10273

[35] T. Lampe, A. Abdolmaleki, S. Bechtle, S. H. Huang, J. T. Springenberg, M. Bloesch, O. Groth, R. Hafner, T. Hertweck, M. Neunert, M. Wulfmeier, J. Zhang, F. Nori, N. Heess, and M. Riedmiller, "Mastering Stacking of Diverse Shapes with Large-Scale Iterative Reinforcement Learning on Real Robots," Dec. 2023, arXiv:2312.11374 [cs]. [Online]. Available: http://arxiv.org/abs/2312.11374

[36] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. Gonzalez Arenas, K. Gopalakrishnan, K. Han,

K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," 2023. [Online]. Available: https://robotics-transformer2.github.io

[37] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[38] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[39] ——, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[40] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement Learning with Augmented Data," Nov. 2020, arXiv:2004.14990 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2004.14990

[41] A. X. Lee, C. M. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, *et al.*, "Beyond pick-and-place: Tackling robotic stacking of diverse shapes," in *5th Annual Conference on Robot Learning*, 2021.

[42] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn, "Vision-Based Manipulators Need to Also See from Their Hands," Mar. 2022, arXiv:2203.12677 [cs]. [Online]. Available: http://arxiv.org/abs/2203.12677

[43] D. Pfrommer, T. T. C. K. Zhang, S. Tu, and N. Matni, "TaSIL: Taylor Series Imitation Learning," Jan. 2023, arXiv:2205.14812 [cs]. [Online]. Available: http://arxiv.org/abs/2205.14812

[44] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "DART: Noise Injection for Robust Imitation Learning," Oct. 2017, arXiv:1703.09327 [cs]. [Online]. Available: http://arxiv.org/abs/1703.09327

[45] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal conditioned imitation learning using score-based diffusion policies," in *Robotics: Science and Systems*, 2023.

[46] A. Block, A. Jadbabaie, and D. Pfrommer, "Provable Guarantees for Generative Behavior Cloning: Bridging Low-Level Stability and High-Level Behavior," *37th Conference on Neural Information Processing*, Sept. 2023.

[47] Y. Tian, K. D. Willis, B. A. Omari, J. Luo, P. Ma, Y. Li, F. Javid, E. Gu, J. Jacob, S. Sueda, *et al.*, "Asap: Automated sequence planning for complex robotic assembly with physical feasibility," *arXiv preprint arXiv:2309.16909*, 2023.

[48] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive Supercomputing on 40,000 Cores for Machine Learning and Data Analysis," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, Sept. 2018, pp. 1–6, iSSN: 2377-6943. [Online]. Available: https://ieeexplore.ieee.org/document/8547629