

A Task Details

In this section, we give additional information on the tasks studied in this work. We give verbal descriptions in Appx. A.1, definitions of data scales in Appx. A.2, and details on the evaluation procedures in Appx. A.3.

A.1 Task Descriptions

- *FoldSock*. Fold a sock (with random configuration) neatly in half.
- *HangOvenMitt*. Hang an oven mitt (with random position and orientation) on a hook (fixed position).
- *HangTape*. Hang a roll of masking tape (with random initial position) on a hook (fixed position).
- *NutInsertion*. Insert a plastic nut (with random initial position) on a peg (fixed position).
- *Square*: Insert a square nut on a square peg (from [24]).
- *SoupInBasket*: Place a small soup can into a basket (from [29]).
- *BookInCaddy*: Place a book into a narrow book caddy (from [29]).
- *StackBowls*: Stack two bowls together and place both on a plate (from [29]).
- *RedMugOnPlate*: Put a red mug on a specific plate (from [29]).

We illustrate initial and final states of our real-world and simulation tasks in Fig. 8. We include an illustration of initial state distributions, sample initial and successful states, and sample camera observations for the NutInsertion and HangTape tasks in Fig. 9.

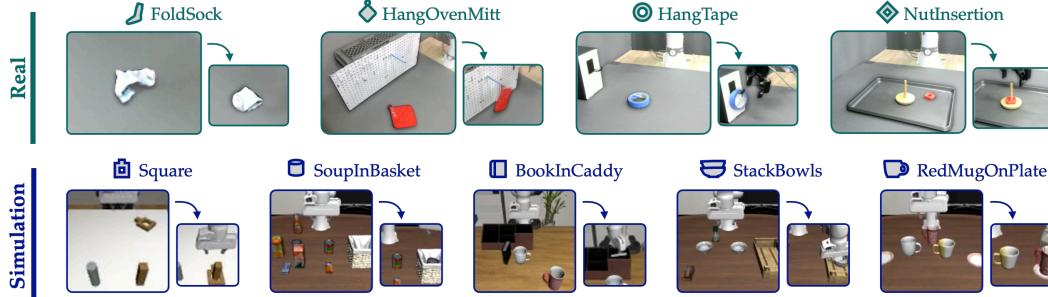


Figure 8: Initial and final success states in 4 real-world environments (top) and 5 simulation environments (bottom).

A.2 Data Scale Definitions

For concision, and to focus on trends, we abbreviate data scales (i.e., number of demonstrations) as low (\downarrow), medium (\diamond), and high (\uparrow) for each of human demonstrations (H) and autonomous rollouts (A). Due to the fact that tasks vary widely in difficulty, the absolute value of demonstrations for each data scale varies per task. We include these values in Table 1.

Env	$\downarrow H$	$\diamond H$	$\downarrow A$	$\diamond A$	$\uparrow A$
FoldSock	100	250	—	—	—
HangOvenMitt	200	500	—	—	—
HangTape	20	50	40	100	320
NutInsertion	50	100	100	250	—
Square	10	50	100	200	500
SoupInBasket	2	5	50	—	100
BookInCaddy	2	5	50	—	100
StackBowls	2	5	50	—	100
RedMugOnPlate	2	5	50	—	100

Table 1: Legend of data scales for each environment.

Example. To generate the training set for the $\downarrow H + \downarrow A$ setting on the NutInsertion task, we do the following:

- Collect 50 human demonstrations from randomly sampled initial states.
- Train an initial policy on the human demonstrations to convergence (approximately 47% success rate).

Task	Initial State Distribution	Sample Initial State	Successful State	Wrist Camera Observation
HangTape				

Figure 9: For the HangTape and NutInsertion tasks, we include scene images depicting the initial state distribution (using an overlay of initial state samples), a sample initial state, a successful state, and a view of the initial state from the wrist camera’s perspective.

- Collect 100 successful autonomous rollouts (by rolling out the policy over 200 times and filtering out the failures).

A.3 Evaluation Procedure

Unless otherwise specified, all success rates in this work are calculated by uniformly sampling an initial state $s_0 \sim \rho_0$ and rolling out the learned policy under consideration until either a success state is achieved or a maximum time horizon is reached. For all simulation results, we perform 200 trials. For all real results, we perform 100 trials.

A.4 Success Detection and Resets

In this section, we provide additional details and rationales for the success detection and reset pipelines that we used in our real-world tasks. For tasks in simulation, success detection and resets were provided by the environment.

A.4.1 Success Detection

- *FoldSock*. As we found that scripting a sock-foldedness detector based on heuristics like object shape and area produced false positive and false negative rates on the order of 20%, we attempted to train a success classifier using a similar procedure to [18]. We assemble a training dataset of 200 human demonstrations (which are curated to be always successful) and roughly 700 rollouts from the autonomous collection policy (which we hand-label as success or failure). The training set includes 301 successful trajectories and 438 failure trajectories, and we sample from the end from each rollout (last 5 images) to yield images to associate with the success/failure label. We train a ResNet-18-based architecture with a binary classification head. The validation error of the trained classifier is approximately 15%.
- *HangOvenMitt*, *HangTape*, *NutInsertion*. These tasks include bottlenecks which must be reached in order to succeed at the task: hanging an object on a hook or placing an object on a peg. Therefore, successes and failures are easy to separate. For simplicity, we use scripted rules similar to prior work (e.g., [10]). Specifically, we use color thresholds at pixels located at these bottlenecks, coupled with the condition that the gripper must be open for five steps prior to success. This ensures that the agent has placed the relevant object at the bottleneck in question. We manually verify that the error rate of this detection scheme is near-zero. While we could in principle train classifiers to learn the boundary between success and failure, our higher-level message is that environment challenges like success detection can be a bottleneck for realistic tasks like *FoldSock*, and can influence task design to make tasks more constrained such that success and failure are easy to detect. In §4, we set aside environment challenges (i.e., assume

that robust success detection is available) in order to study whether autonomous IL can reduce human supervision challenges.

A.4.2 Resets

In our study, we use object-centric primitives of various complexity to perform resets. Instrumenting environments with hand-crafted primitives, physical reset mechanisms, or requiring humans to perform resets is a common technique in real-world reinforcement learning [20, 21]. As we illustrate in §3, the human effort of environment design (e.g., by instrumenting the environment to make reset primitives possible) remains when we utilize autonomous IL methods, and these can get more involved as we move towards more useful and complex tasks.

- *FoldSock*. We reset the scene by flinging the sock: locating the sock using a segmentation pipeline (GroundingDINO [32] + FastSAM [33]), picking it up using a top-down grasp, bringing it to the center of the workspace, and executing a fling primitive to randomize its configuration for the next episode.
- *HangOvenMitt*. The final state of the mitt has two cases—in the case of success, the mitt is hanging and the mitt can be pulled off the hook by replaying a pre-recorded trajectory; in the case of failure, the mitt is pulled back to a reachable location via a string attached to the robot—and in both cases, the mitt’s location is then randomized using a parameterized pick-and-place primitive.
- *HangTape*. We follow a similar procedure as in HangOvenMitt: if the tape is on the hook (i.e., the previous episode was successful), we replay a pre-recorded trajectory to pull it off of the hook. Otherwise, we detect the location of the tape using a simple color mask and execute a pick-and-place primitive to randomize its initial location for the next episode.
- *NutInsertion*. We once again utilize the fact that the final state of the previous episode is either a success, for which the nut can be removed from the peg using a pre-recorded trajectory, or a failure, for which the nut’s location can be randomized using a pick-and-place primitive.

B Analyzing Human Supervision: Additional Results

In this section, we provide further details on the results in §4 of the main text. In Appx. B.1, we ablate the choice of training from scratch on human-autonomous mixtures (the recipe used in all experiments in the main text). We also provide additional details regarding training with different data weights (Appx. B.2), data scales (Appx. B.3), policy class (Appx. B.3.1), number of rounds (Appx. B.4), and novelty-based reweighting (Appx. B.5), active learning from failures (Appx. B.6), and offline RL (Appx. B.7). While experiments in the main text focus on autonomous data collected in-distribution, we provide additional experiments in Appx. B.8 on training with autonomous data collected from out-of-distribution (OOD) scenarios. Finally, we provide qualitative examples of human and autonomous trajectory distributions in Appx. B.9.

B.1 Training from Scratch vs. Fine-tuning

All of the models trained on human-autonomous data mixtures in §4 are trained from scratch until convergence. In this subsection, we justify this choice by comparing training from scratch to methods involving fine-tuning.

Specifically, we focus on a single round of autonomous collection for the Square task in simulation. Unless otherwise specified, each model is trained on a mixture of 50% autonomous, 50% human data. We compare the following training recipes:

- *Scratch*: Train a new model from scratch on the human-autonomous mixture.
- *Fine-tune*: Fine-tune the autonomous policy checkpoint that generated the autonomous data on the human-autonomous mixture.
- *Pre-train Autonomous + Fine-tune*: Pre-train a policy from scratch on the autonomous data only, and then fine-tune on the human-autonomous mixture.
- *Scratch Add*: Directly aggregate human and auto data in one dataset (no explicit 50-50 sampling), and train from scratch on this dataset.

In Table 2, we find that training from scratch, fine-tuning from the base policy, and training on combined human and auto datasets all perform comparably. In fact, training methods seem to matter much less than

the amount of autonomous data provided. Therefore, for simplicity, we use the *Scratch* training method for all other experiments in the main text.

Method	$\diamond H + \downarrow A$	$\diamond H + \diamond A$	$\diamond H + \uparrow A$
Scratch	69%	61.5%	79.5%
Fine-tune	68.5%	66%	67.5%
Pre-train Auto + Fine-tune	68.5%	69.5%	73.5%
Scratch Add	68.5%	66%	77.5%

Table 2: Comparing different training methods on Square in simulation, for medium amounts of human data ($\diamond H$) but for increasing amounts of autonomous data ($\downarrow A$ to $\diamond A$ to $\uparrow A$). All methods perform equivalently in each data regime.

B.2 Human and Autonomous Data Weights

Our experiments on Data Weights study the impact of relative sampling weights of human-to-autonomous data in the training mixture (i.e., changing `mix`). These experiments keep the amount of autonomous data fixed ($\downarrow A$) and investigate if success rate changes for two scales of human data ($\downarrow H$ and $\diamond H$) at different sampling ratios (75-25, 50-50, 25-75). We include these results in table form in [Table 3](#) and [Table 4](#). We find that changing the training data weights has almost no impact for a given data scale. This is line with expectations from prior work when using importance weighted objectives with highly expressive models [34]. Guided by these results, we use the simple training from scratch setting with 50-50 human-autonomous mixtures for the remaining experiments in §4.

Env	$\downarrow H$ 75-25	$\downarrow H$ 50-50	$\downarrow H$ 25-75	$\diamond H$ 75-25	$\diamond H$ 50-50	$\diamond H$ 25-75
Square	15.5%	22%	21%	37.5%	38.5%	41%
SoupInBasket	37%	33.5%	40.5%	72%	74%	77.5%
BookInCaddy	28.5%	30.5%	36.5%	58%	60%	62%
StackBowls	53%	59.5%	57%	69.5%	76%	68.5%
RedMugOnPlate	80%	80%	83%	82.5%	81.5%	82%

Table 3: Different training weightings of human to autonomous data in simulation have negligible effects.

Env	$\downarrow H$ 75-25	$\downarrow H$ 50-50	$\downarrow H$ 25-75
HangTape	47%	55%	57%
NutInsertion	59%	58%	48%

Table 4: Different training weightings of human to autonomous data in real have negligible effects.

B.3 Human and Autonomous Data Scales

Our experiments on Data Scales ([Fig. 4](#)) use a 50-50 mixture and examine how success rate is impacted by the scale of initial human data and the ratio of human to autonomous data. We include the results in table form in [Table 5](#). Including some amount of autonomous data tends to have mild positive effects in most cases, though these effects generally saturate as autonomous data scales. Increasing the scale of human data generally has a stronger effect than adding autonomous data.

B.3.1 Human and Autonomous Data Scales under Different Policy Classes

In this section, we provide additional results on Data Scales using a 50-50 mixture, keeping the task the same but testing two different policy classes: Diffusion Policy (DP) [5] and Action Chunking with Transformers (ACT) [4]. Both methods are capable of modeling diverse action distribution modes. While ACT underperforms DP in this task, the effects on success rate when re-training with different scales of autonomous data are largely similar: there is mild improvement which appears to plateau. The compatible results on ACT and Diffusion Policy suggest that our observations are not unique to the policy class.

Env	$\downarrow H$	$\downarrow H + \downarrow A$	$\downarrow H + \uparrow A$	$\diamond H$	$\diamond H + \downarrow A$	$\diamond H + \uparrow A$
Square	15.5%	22%	16%	44.5%	38.5%	43.5%
SoupInBasket	16.5%	33.5%	45.5%	54.5%	74%	83%
BookInCaddy	40.5%	30.5%	33%	51.5%	60%	61.5%
StackBowls	50.5%	59.5%	54%	83%	76%	81.5%
RedMugOnPlate	58%	80%	82.5%	79%	81.5%	86%
HangTape	44%	55%	46%	80%	80%	86%
NutInsertion	44%	58%	64%	53%	61%	—

Table 5: Scales of human data compared to autonomous data for 50-50 co-training on various simulation (top) and real (bottom) environments. More autonomous data often helps, but having more human data generally has a stronger effect.

Env	Method	$\downarrow H$	$\downarrow H + \downarrow A$	$\downarrow H + \uparrow A$	$\diamond H$	$\diamond H + \downarrow A$	$\diamond H + \uparrow A$
HangTape	DP	44%	55%	48%	80%	80%	86%
HangTape	ACT	26%	32%	27%	32%	44%	40%

Table 6: Scales of human data to autonomous data for 50-50 co-training on the HangTape environment when varying the policy class between Diffusion Policy [5] and Action Chunking with Transformers [4]. Similar trends exist between the two policy classes: autonomous data often helps, but no more than additional human data, and the improvement quickly plateaus.

We choose Diffusion Policy for the remainder of experiments in this work because it is a state-of-the-art IL method and has the same policy class as a state-of-the-art offline RL method, IDQL, which we look at in §4.4.

B.4 Multiple Collection Rounds

Our experiments on Multiple Collection Rounds (Fig. 5) measure if any positive effects of autonomous data continue over multiple iterations. Specifically, we replace the autonomous data in the training mixture with the latest round of autonomous data collection, and re-train the model from scratch. The amount of autonomous data is kept constant at each round ($\diamond A$; $\uparrow A$ for LIBERO tasks). We investigate the effects of multiple collection rounds at multiple scales of human data ($\downarrow H$ and $\diamond H$) in simulation and at the $\downarrow H$ scale in real. We present the results in table form in Table 7 and Table 9, generally observing plateaus in performance after an initial improvement in the first iteration. Interestingly, in the Square task, we observe a slight *decrease* in performance. Unlike the LIBERO tasks, Square contains a more challenging bottleneck state, and we hypothesize that subtle variations in the action distributions over multiple rounds of autonomous data collection and training may amplify this challenge. As evidence, in Table 8, we examine the “staged” success rate in Square over multiple iterations: note that the subtask for “moving the square” increases in success rate while the full task (which includes the insertion bottleneck) decreases in success rate.

Env	Base	Round 1 ($\diamond A$)	Round 2 ($\diamond A$)	Round 3 ($\diamond A$)	Round 4 ($\diamond A$)
Square ($\downarrow H$)	15.5%	17%	13%	21%	18.5
Square ($\diamond H$)	44.5%	38.5%	36%	35%	35%
SoupInBasket ($\downarrow H$)	16.5%	45.5%	60%	78%	—
SoupInBasket ($\diamond H$)	54.5%	84%	82.5%	82%	—
BookInCaddy ($\downarrow H$)	40.5%	40%	37.5%	44%	—
BookInCaddy ($\diamond H$)	51.5%	64%	74.3%	72%	—

Table 7: Multiple Rounds of autonomous collection using medium autonomous data ($\diamond A$) and training in simulation ($\downarrow H$ and $\diamond H$). We see either saturating increases or decreases in performance.

Stage	Base	Round 1 ($\diamond A$)	Round 2 ($\diamond A$)	Round 3 ($\diamond A$)	Round 4 ($\diamond A$)
Moves Square	67.5%	99.5%	100%	100%	94.5%
Full Success	44.5%	38.5%	36%	35%	35%

Table 8: Multiple Rounds of autonomous collection in Square ($\downarrow H$), illustrating the success rate for an intermediate stage (moving the square) and the full task.

Env	Base	Round 1 ($\diamond A$)	Round 2 ($\diamond A$)
HangTape ($\downarrow H$)	44%	55%	50%
NutInsertion ($\downarrow H$)	47%	57%	46%

Table 9: Multiple Rounds of autonomous collection using medium autonomous data ($\diamond A$) in real for HangTape and NutInsertion. We see that even though success rates improve in Round 1, they do not improve in Round 2.

B.5 Novelty-Based Reweighting Strategies

In §4.3, we consider if state novelty can be used as a proxy to extract more useful autonomous data, and form the basis for a sampling weight. In this section, we provide more details on these novelty measures. Given an ensemble of policies $\mathcal{E} = \{\pi_1, \pi_2, \dots, \pi_N\}$, we instantiate two measures of novelty building on ideas from prior work [13, 23, 30].

1. *Action Novelty*: Measure state novelty as proportional to the variance in the mean action predictions. This variance can be measured by an ensemble of policies trained on the same data:

$$\text{ActionNovelty}(s) = \sum_{i=1}^{N_A} \text{Var}_j(\mu_{ji})$$

where μ_j is the mean of the predicted action distribution $\pi_j(s)$ and N_A is the number of action dimensions.

2. *Image Embedding Novelty*: Measure state novelty as proportional to the variance in image embeddings produced by an ensemble of vision encoders (i.e., the encoders from each policy in \mathcal{E}):

$$\text{EmbeddingNovelty}(s) = \sum_{i=1}^{N_h} \text{Var}_j(h_{ji})$$

where $h_j = \text{enc}_j(s)$ (i.e., the embedding from the encoder associated with policy π_j) and N_h is the number of embedding dimensions.

Given a novelty measure, we assign the training weight for state s to be proportional to $\exp(\text{Novelty}(s)/\beta)$ where β is a temperature hyperparameter.

B.6 Active Learning Guided by Failures

In §4.4, we examine if we can target data collection by utilizing initial states of failed autonomous rollouts. We provide performance trends for policies trained on different amounts of additional human and autonomous data, both targeted and random, when added to the initial $\downarrow H$ dataset (10 demonstrations in the case of Square and 20 demonstrations in the case of HangTape). We see consistently that random and targeted human data collection outperforms the same amount of random and autonomous data, and also has a higher slope. In Square, there appears to be a dropoff in the relative improvement of targeted human data above random human data. Neither random nor targeted autonomous data improve upon the initial policy in Square, and the improvements from autonomous data are mild in HangTape.

B.7 Offline RL with Autonomous Successes and Failures

One can argue that failure data has more to offer than just initial states as in §4.4. We turn to offline RL to learn *directly* from both successful and failure examples: modifying F to accept both successes and failures, and setting B to be an offline RL algorithm. We use Implicit Diffusion Q-Learning (IDQL) [34], a state of the art offline RL algorithm, that uses both success and failure data to learn a Q-function expectile, and then uses this to sample high Q-value actions from a generative actor.

To not introduce even more environment challenges, we use sparse rewards provided by the same success detection function. We use Diffusion Policy as the generative actor, resampling actions under the Q-function at each time step. In Fig. 10, we compare IDQL to DP trained on successful autonomous data (BC) and a mixture of successful and failure data (SUB). We observe that incorporating failures through IDQL does not outperform naïve autonomous IL, and only slightly outperforms the suboptimal autonomous IL trained on success and failure data. This could be because IDQL struggles to learn a good Q-function estimate from such a small amount of data and such a high dimensional state space (images). These findings are consistent with prior offline RL results in practice [35].

B.8 Training on Out-of-Distribution Autonomous Successes

The experiments in the main text focus on training with autonomous data that is collected from *in-distribution* initial states (i.e., initial states are sampled from ρ_0 uniformly, or in the case of the active learning experiments, a reweighted version of ρ_0). In this section, we examine possible benefits from training on successful autonomous data from out-of-distribution (OOD) scenarios. More specifically, we generate the autonomous data by rolling out the initial policy from a new initial distribution ρ'_0 and collect autonomous successes which are the result of the policy generalizing to the new distribution.

In Table 10, we examine the impact on success rates when adding OOD autonomous data in the HangTape task. Specifically, we collect OOD autonomous data where one of two factors is varied compared to the initial distribution: the object (i.e., the tape is changed to a different roll of tape with a different color) and the distribution of initial object positions (i.e., the initial locations are sampled at an expanded outer boundary of the original distribution). When adding 50 successful autonomous rollouts from either of these OOD conditions to 50 in-distribution human demonstrations, we find positive impacts both in-distribution and in the OOD conditions. We see a similar trend in Table 11 on the NutInsertion task, where we collect autonomous data in OOD initial positions (i.e., the initial locations are from an expanded outer boundary) and find that both in-distribution and OOD performance improves.

These insights suggest that OOD autonomous data—i.e., successes that are the result of generalization in the initial policy—may be valuable, at the cost of potentially increasing environment design effort to change the initial state distribution of the environment.

Data Mixture	Success (ID)	Success (OOD Position)	Success (OOD Object)
50 H (ID)	80%	13%	27%
50 H (ID) + 50 A (OOD Position)	90%	23%	—
50 H (ID) + 50 A (OOD Object)	83%	—	51%

Table 10: Success rates both in-distribution (ID) and out-of-distribution (OOD) for policies trained on mixtures of in-distribution human data and OOD autonomous data on the HangTape task.

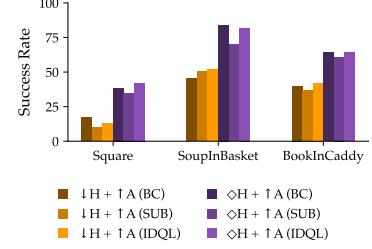


Figure 10: Offline RL results, comparing IDQL trained on mixed success and failure data to the naïve autonomous IL strategy (BC), and a suboptimal (SUB) version of naïve autonomous IL trained on both successes and failures. IDQL matches BC and slightly beats SUB.

Data Mixture	Success (ID)	Success (OOD Object)
50 H (ID)	44%	40%
50 H (ID) + 50 (OOD Object)	52%	50%

Table 11: Success rates both in-distribution (ID) and out-of-distribution (OOD) for policies trained on mixtures of in-distribution human data and OOD autonomous data on the NutInsertion task.

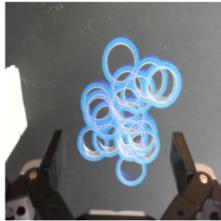
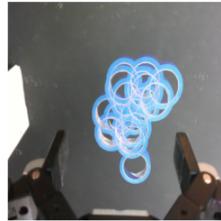
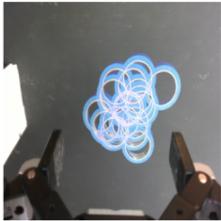
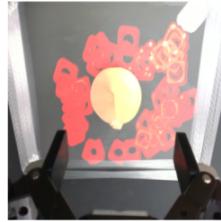
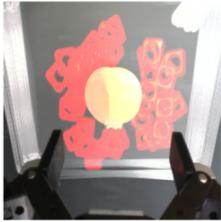
Task	Initial States of Human Demonstrations	Sample Initial States of Successful Autonomous Rollouts	Sample Initial States of Successful Rollouts After 50-50 Co-training
HangTape			
			

Figure 11: Comparison of initial states from human demonstrations, autonomous rollouts, and rollouts from policies co-trained on human and autonomous data. The left column shows, from the wrist camera perspective, superimposed initial states from human data. These initial states are sampled from the initial state distribution of the task, and correspond to the data for the $\downarrow H$ setting. Specifically, this corresponds to 20 demonstrations for HangTape and 50 demonstrations for NutInsertion. In the middle column, we illustrate initial states from sampled successful rollouts of the autonomous collection policy (trained on the human data). In the right column, we illustrate initial states from successful evaluation rollouts from the $\downarrow H + \downarrow A$ policy, which is co-trained with a 50-50 mixture of human and autonomous data. Note that, for visualization purposes, the middle and right columns show same number of sampled successful initial states as there are demonstrations in the left column.

B.9 Qualitative Examples of Human and Autonomous Data Distributions

In this section, we take a qualitative look at the data distributions of teleoperated human demonstrations compared to autonomous rollouts from policies trained on the human data. We additionally compare the distribution of rollouts from policies co-trained on human and autonomous data.

Fig. 11 illustrates sample initial state distributions from these three categories. In the left column, we superimpose initial states from human teleoperated demonstrations; these initial states are sampled from the initial state distribution of the task. These correspond to data sources for the $\downarrow H$ settings of HangTape and NutInsertion (20 and 50 demonstrations respectively). In the middle column, we sample initial states from *successful* autonomous rollouts from a Diffusion Policy trained on the human data. These policies are used as the autonomous data collection policies. Note that only a random sample of the successful autonomous data is shown for visualization purposes (a sample of 20 and 50 for HangTape and NutInsertion respectively). Finally, in the right column, we show sample initial states from successful rollouts of a Diffusion Policies co-trained on the human data and autonomous data (with 50-50 data weights). These policies correspond to the $\downarrow H + \downarrow A$ settings from §4.2.

In Fig. 12, we similarly illustrate trajectories (end-effector positions) for human demonstrations, sampled successful autonomous rollouts, and sampled successful rollouts of policies trained on human+autonomous data. From Fig. 11, we see a narrowing effect in the distribution of successful initial states, which is more pronounced in the HangTape environment. The policy trained on human demonstrations learns to interpolate between initial locations of the tape that are represented in the human data, especially towards the center of the distribution. When the policy is re-trained with a mixture of human data and autonomous data, the spread in the distribution of initial states appears to get reduced. However, note that we observe mild overall increases in success rate from autonomous data, and so this is likely due to the policy becoming slightly more robust towards the center of the distribution.

In Fig. 12, we observe an increased homogenization in the successful trajectory paths. This extends beyond just the initial state distributions; note that in both the HangTape and the NutInsertion task, the segments of the trajectories before grasping the object are straighter and less diverse than the corresponding segments

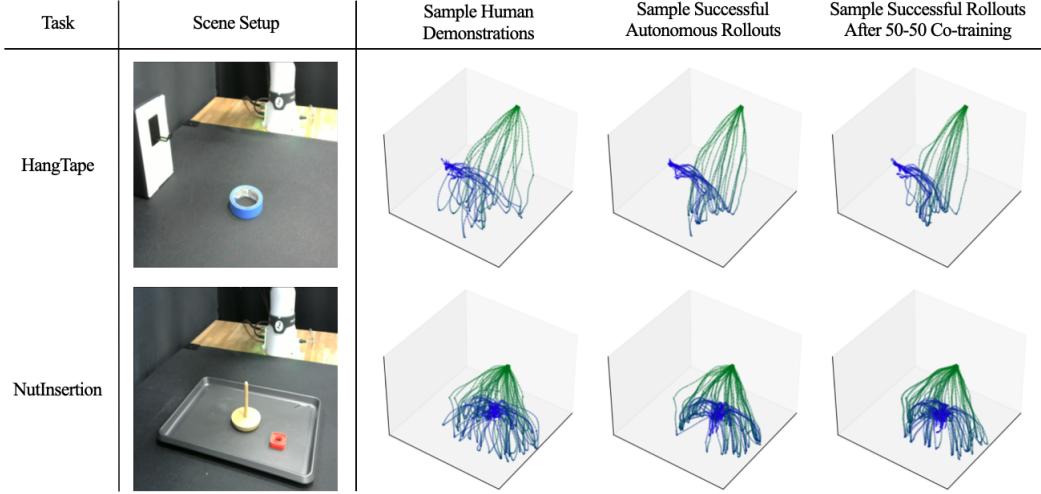


Figure 12: Illustration of trajectories (as 3D end-effector paths) from various policies, with green representing the start of the trajectory and blue representing the end. For reference, we show the scene setup with a sample initial object location in the leftmost column. The second column illustrates human teleoperated demonstration trajectories. The third column illustrates successful autonomous rollouts from a Diffusion Policy trained on the human demonstrations ($\downarrow H$). The fourth column illustrates successful rollouts from a Diffusion Policy co-trained on human data and autonomous data ($\downarrow H + \downarrow A$).

in the human data. Additionally, note that the strategies used post-grasp to place the object at its final location (hanging the tape on the hook in the case of HangTape, or placing the nut on the peg in the case of NutInsertion) become more consistent in the autonomous data (as well as the policy co-trained on autonomous data) compared to the human demonstrations.

C Training Hyperparameters

Diffusion Architecture	Conv1D UNet
Prediction Horizon	16
Observation History	2
Num Action	8
Kernel Size	5
Num Groups	8
Step Embedding Dim	256
UNet Down Dims	[256, 512, 1024]
Num Diffusion Steps	100
Num Inference Steps	10
Inference Scheduler	DDIM
Observation Input	FiLM
Image Encoder	ResNet-18
Image Embedding Dim	256
Proprioception	yes

Table 12: Hyperparameters for Diffusion Policy, shared for all simulation experiments.

For all simulation experiments, we train using Diffusion Policy [5] with the hyperparameters in [Table 12](#) and [Table 13](#). Our real-world experiments use the same hyperparameters, except with an observation history of 1, a step embedding dimension of 128, and 2000 warmup steps. We train policies for the HangTape task

Training Steps	500K
Batch Size	64
Optimizer	AdamW
Learning Rate	1e-4
Weight Decay	1e-6
Learning Rate Schedule	Cosine Decay
Linear Warmup Steps	1000

Table 13: Training Hyperparameters, shared for all simulation experiments.

for 400K steps and policies for the NutInsertion task for 500K steps. For our ACT experiments, we use the default hyperparameters from [4] except with a chunk size of 16. We execute 8 actions for each inference step at execution time. For the HangTape task, we train policies with 20 human demonstrations for 200K steps and policies with 50 human demonstrations for 400K steps based on model selection between 200K, 400K, and 500K steps.