Data-Efficient Robot Learning in Deployment

Jane E. Doe
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley United States
janedoe@berkeley.edu

Figure 1: *Diverse initial states:* We study learning in deployment with wide initial-state distributions. *Concentrated failures:* Failures are not spread evenly across the initial-state space. *Targeted collection:* Choosing each round's initial states from observed failures, rather than sampling them, spends operator time where it is needed.

Abstract: Learning from human demonstrations is a reliable way to teach robots new tasks, but the returns on additional demonstrations often diminish as the policy improves. Continued improvement can instead come from data collected in supervised deployment, where an operator places the objects for each episode and intervenes when the policy fails. We ask how to maximize the return on operator and robot time for high-precision manipulation under the wide initial-state distributions of realistic deployment. We observe that failures are often unevenly distributed across the initial-state space, so uniform collection spends much of the operator's time on states the policy already handles. In response, **Mulligan** makes the initial-state distribution a decision by steering each round's collection toward the states where the policy fails. To further improve data efficiency, we augment interactive imitation learning with a value function trained on all data, including the failures that imitation otherwise discards. Across three real-world tasks evaluated on 2,850 held-out, blinded episodes and two simulated tasks, **Mulligan** outperforms uniform sampling of initial states at matched supervised collection budgets, and combined with value-based action selection, HiL-IDQL+**Mulligan**, improves final real-task success by 14–34 percentage points over HG-DAgger. With operator interventions, the robot completes 95–100% of collection episodes in every round, so improvement does not come at the cost of completing tasks.

1 Introduction

Human demonstrations are the usual starting point for teaching a robot a new manipulation task [1, 2, 3, 4, 5], but the returns on additional demonstrations often diminish as success rate improves, especially for tasks requiring high precision [6, 7, 8, 9]. A common next step is to collect corrections with a human in the loop (HiL), where an operator places the objects for each episode, takes control when the policy goes off track, and the corrections join the training set [10, 11, 12, 13]. Recent

systems show this reliably achieves high success rates on physical hardware [14, 13, 15], but every episode still costs operator time. The cost is easier to bear in supervised deployment, where an operator already oversees the robot on real tasks and its rollouts arrive as a byproduct. We study the batch-online form of this loop [16], which retrains the policy between rounds rather than during collection, and ask how to get the most improvement out of each supervised episode.

Several lines of work address parts of this question. Interactive imitation learning trains any policy class by behavior cloning (BC) on the corrections, but imitation cannot learn from the failed rollouts that deployment yields as a byproduct [12, 13]. Real-world RL learns from every rollout, failures included, through a value function, but gathers those rollouts by exploring online on the hardware [17, 18, 19, 20, 21, 22, 23, 24] and has shown its best results under narrow initial-state distributions [17, 18, 19] (ranges compiled in Sec. K.4). Batch-online systems [16] combine corrections with a value function, training it on the deployed rollouts as well, under a policy updated only between rounds and at the wide initial-state distributions of real deployment [14, 15]. The lines differ in the policy class and the learning update. The most recent bring corrections and RL to vision-language-action (VLA) policies at scale rather than asking how much improvement each supervised episode can yield.

Our key observation is that failures are often unevenly distributed across the initial-state space. In the example in Fig. 1 (middle), 95% of a policy's failures fall in 25% of the initial states even though its overall success rate is 86%. Raising the success rate means correcting those failures where they happen. Under uniform collection, the policy makes fewer mistakes proportionally as it improves, and the operator becomes an increasingly passive observer instead of providing useful corrections. Yet the operator who decides when to intervene also places the objects at the start of every episode, so the initial-state distribution is already a decision rather than a fixed part of the task, and it can be made deliberately. The choice is independent of the policy class and the learning update, which the methods above vary, so it combines with any of them. The question is then how to find those states without searching the whole space.

RB: **Mulligan** collects data in rounds, with two choices per round: which initial states to visit, and what the operator records there. **Mulligan** finds them with the outcomes each round already produces (Fig. 1, right, and Sec. 3.1). The first round covers the initial-state space evenly and yields the first evidence of where the policy fails. Each later round balances retrying the initial states where the previous one failed with exploring the rest of the space for new failures. At a retried initial state, a correction begins where the policy went off track, so when the mistake is big, it teaches recovery rather than what the policy should have done. Instead, the operator can record a full demonstration from the same initial state, a *counterfactual demonstration* that places the missing behavior in the policy's support. **Mulligan** only decides what each round collects, so we evaluate it with two learners, imitation alone and imitation with a value function trained on all rollouts, failures included, that selects among the policy's actions (Sec. 3.2).

Our main contribution is **Mulligan**, a data-collection strategy for batch-online imitation and reinforcement learning that selects each round's initial states based on the policy's observed performance, yielding more improvement from the same supervised budget. Across three real-world manipulation tasks, evaluated on 2,850 held-out, blinded episodes, and two simulated tasks, **Mulligan** outperforms uniform sampling of initial states at matched supervised collection budgets. Combined with value-based action selection, HiL-IDQL+**Mulligan** improves final-round success on the real tasks by 14–34 percentage points over HG-DAgger [11] while also completing more tasks per robot minute (Sec. 4). With an operator intervening, the robot completes 95–100% of collection episodes in every round, so learning does not come at the cost of completing tasks during collection.

2 Related Work

CF: Generally, when writing the related work section, it's nice to frame things as positive w.r.t. this work rather than negative w.r.t. prior works. e.g., instead of "autonomous failures are discarded",

you can say "unlike these works, we leverage both autonomous data and human intervention data". This comment applies throughout the related work section.

Human-in-the-loop robot learning. We build on interactive imitation learning, which collects human corrections under the policy's own state distribution and thereby provides recovery supervision exactly where the policy drifts [10, 11]. Work in this line has studied when the human should intervene, by expert judgment [11] or estimated uncertainty [25, 26], and has scaled collection to long-horizon bimanual manipulation with expressive policies [7, 13, 12]. Interventions also carry implicit reward signal [27], and a second family combines human-in-the-loop collection with RL: RLIF [20] derives rewards from intervention signals, HiL-SERL [18] learns from corrections via RLPD [28], and ConRFT [19] and EXPO-FT [21, 29] extend this to finetuning VLAs. We share their aim of learning from the robot's own rollouts, not only the corrections, and update the policy between rounds rather than online during collection. Closest to our setting are RECAP [14] and LWD [15], which improve VLAs over rounds of supervised deployment at scale. These recipes focus on the learning update, e.g., advantage-conditioned training on accumulated experience in RECAP. **Mulligan** instead decides which initial states each round's episodes begin from, a choice orthogonal to the learning update. Here, we evaluate it with imitation alone and with a value function trained on all rollouts, and it would be compatible with recipes such as RECAP and LWD.

Real-world RL and autonomous self-improvement. Reinforcement learning has trained policies on physical robots since early policy search over motor skills [30, 31] and end-to-end visuomotor learning from pixels [32, 33], and recent systems make online training on the robot sample efficient [17, 34, 35, 22, 21], with their best results under narrow, centimeter-scale initial-state randomization [17, 18, 19] (ranges compiled in Sec. K.4). Within such a range, a handful of episodes reach every initial state, so where each episode starts carries little weight. Over the wide ranges we study, a round's budget covers a fraction of the space, and the choice of initial states becomes consequential. A related line improves robots from their own experience in batches, without a human in the loop [36, 37, 38], and Dong et al. [16] show that a Q-function with best-of-N policy extraction improves policies through autonomous batch-online interaction. We adopt their value-learning and extraction design, since best-of-N improves a generative policy without backpropagating through its sampling chain (Sec. K.2), and add the operator's corrections and counterfactual demonstrations to the data the value function learns from. Sec. K extends this discussion to real-world RL systems and to RL finetuning of expressive policies, the family our best-of-N extraction belongs to.

Choosing where episodes begin. Where episodes begin has been a design choice in RL since restart distributions were used to direct policy improvement [39]. Curricula move initial states toward intermediate difficulty, expanding outward from the goal [40], generating goals of suitable difficulty [41], resetting to states along demonstrations [42, 43, 44, 45, 46] or a guide policy's rollouts [47], or returning to stored states and replaying the levels with the most to teach alongside fresh ones [48, 49]; Narvekar et al. [50] and Portelas et al. [51] survey the area. Hard-example mining concentrates training on what a model gets wrong within a fixed dataset [52]. For real robots, recent work chooses what to collect by which environmental factors to vary [53], which failure conditions a model predicts [54], where the policy is uncertain [55], which failure states its executions reveal [56, 57], which perturbations to apply during a demonstration [58], or in what order a human demonstrates [59]. **Mulligan** moves the initial-state choice to a supervised deployment, where the operator places each initial state, and mixes retried and fresh states as prioritized level replay does. The operator also changes what the choice can reach. Since a human can complete the task from any state, collection can go to the hardest states found so far instead of a band of intermediate difficulty, and a counterfactual demonstration records the missing behavior there instead of waiting for the policy to discover it.

Algorithm 1 The supervised improvement loop, with **Mulligan** choosing the initial states in the Analyze step. HG-DAgger skips the critic and deploys π_θ alone.

$\mathcal{D}_h, \mathcal{D}_{\text{all}} \leftarrow \mathcal{D}_0$, base demonstrations at M initial states drawn from a scrambled Sobol sequence over $\mathcal{S}_0$
for round $r = 1, \ldots, R$ **do**
    Train: actor π_θ by BC on $\mathcal{D}_h$; critic (Q_ψ, V_ξ) on $\mathcal{D}_{\text{all}}$ (Sec. 3.2)
    Evaluate: deploy π_θ with Q_ψ-reranking (eq. (1)) for K episodes; add data to $\mathcal{D}_{\text{all}}$
    Analyze: choose the next M initial states with SELECTINITIALSTATES (Alg. 2)
    Collect: M supervised episodes, each counterfactual demonstration replacing the last coverage-fill initial state; add human frames of successful episodes to $\mathcal{D}_h$ and all frames to $\mathcal{D}_{\text{all}}$
end for

3 Mulligan: Performance-Guided Collection for Supervised Deployment

Mulligan is a collection strategy. It decides where each round's episodes start and what the operator records there (Sec. 3.1). We pair it with two learners. HG-DAgger [11] behavior-clones the human data. HiL-IDQL, our human-in-the-loop variant of IDQL [60], trains the same actor and, with decoupled actor and critic learning, a critic on all of the collected data, including the robot's failures, which reranks the actor's samples at deployment (Sec. 3.2). Alg. 1 gives the loop both learners run. We first define the setting, the notation, and the learning algorithms we build on.

Problem setup. We model each task as a Markov decision process $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics P, a sparse reward r that is 1 when the task is completed and 0 otherwise, discount γ, and an initial-state distribution ρ_0, uniform over a bounded range of object poses $\mathcal{S}_0$. The goal is a policy π_θ that maximizes autonomous success rate on initial states drawn from ρ_0. Physical deployment favors a stable and performant policy rather than one that changes continually through online exploration, so we keep the policy's parameters fixed within a round and update them offline between rounds, which also lets every accumulated frame be reused. This is a *human-in-the-loop* (HiL) *batch-online RL* setting that proceeds in rounds under the supervision of a human *operator*. The operator places the objects at the start of each episode, watches the policy act, and may take over control to provide corrections. Before the first round, the operator records a base set $\mathcal{D}_0$ of demonstrations. In each round, a policy π_θ with fixed parameters during rollout is deployed for a budget of M episodes with the operator present throughout, and all transitions are stored. We then re-train the models on all accumulated data and redeploy. The boundary between rounds is a natural point to analyze the accumulated data and inform the next round's collection. Because the operator places the objects, that collection can start from any initial state in $\mathcal{S}_0$ rather than requiring states to be sampled from ρ_0, and choosing which ones is the central lever of our method.

Data and notation. Supervised deployment produces four kinds of data. The operator provides *demonstrations*, teleoperated completions of the whole task, which include $\mathcal{D}_0$ and the counterfactual demonstrations of Sec. 3.1, and *corrections*, in which the operator takes over mid-episode. The robot contributes its own *autonomous successes* and *autonomous failures*. Beyond the reward, training needs only one label, a per-frame source tag (human or policy). $\mathcal{D}_h$ denotes the human-controlled frames of successful episodes, which covers demonstrations, counterfactuals, and the human segments of corrections, and $\mathcal{D}_{\text{all}}$ denotes every collected frame, whatever its source or outcome.

Learning algorithms. The policy $\pi_\theta(\mathbf{a} \mid s)$ is a Diffusion Policy [1], a generative model over *action chunks* $\mathbf{a} = (a_t, \ldots, a_{t+H-1})$, of which the first $h \leq H$ actions are executed before the policy is queried again. Behavior cloning (BC) fits π_θ to a dataset of state and chunk pairs by maximum likelihood, which for a diffusion model is the denoising loss. Implicit Q-learning (IQL) [61] learns a critic Q_ψ and a value function V_ξ from a dataset without sampling actions from the policy. V_ξ is

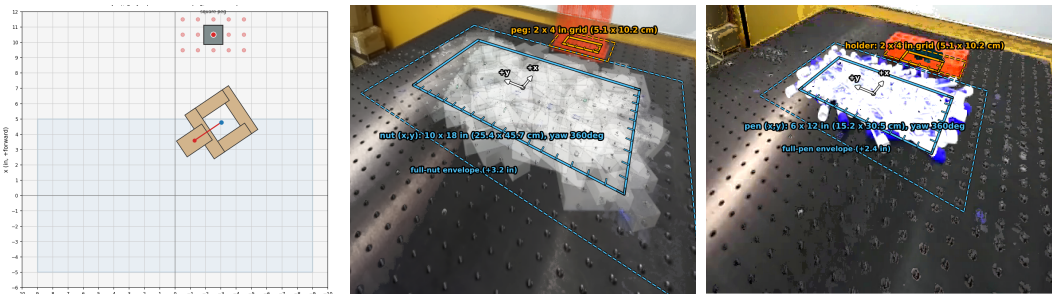

Figure 2: Left: THREAD NUT reset-helper card, the per-episode schematic the operator places objects from (all three tasks in Sec. F.4). Middle and right: initial-state ranges for THREAD NUT (nut pose in white, peg location in red) and INSERT MARKER (pen pose in white/blue, holder location in red), visualized as an overlay of sampled reset poses with randomization-range rectangles and their physical extents annotated. The ROUTE CABLE overlay is in Fig. 1, left.

fit to Q_ψ by expectile regression, minimizing $\mathbb{E}\big[\,|\tau - \mathbb{1}[u < 0]|\,u^2\,\big]$ with $u = Q_\psi(s, \mathbf{a}) - V_\xi(s)$. For $\tau > 0.5$, the value tracks the better actions in the data. Q_ψ is then regressed onto the target $r + \gamma V_\xi(s')$ for each transition $(s, \mathbf{a}, r, s')$ in the data. Implicit diffusion Q-learning (IDQL) [60] extracts an improved policy from such a critic without gradients through the actor. At each state it draws N chunks $\{\mathbf{a}_1, \ldots, \mathbf{a}_N\} \sim \pi_\theta(\cdot \mid s)$ and executes

$$\mathbf{a}_{i^\star}, \qquad i^\star = \arg\max_i Q_\psi(s, \mathbf{a}_i), \tag{1}$$

which we refer to as best-of-N (BoN) sampling and which later work has adopted as a stable improvement operator for generative policies [16, 62, 29].

DY: I think the actor/critic parallel structure should be made explicit here. Right now BC, IQL, and IDQL are introduced sequentially, and the reader has to infer their roles. Suggested organization: "We train the actor and critic separately. The actor ... [BC]. Separately, the critic ... [IQL]. At deployment, IDQL combines the two by using the critic to rerank actor samples."

3.1 Performance-guided collection

Not all of $\mathcal{S}_0$ is equally hard. Fig. 1 (middle) shows a policy that is near-perfect over most of $\mathcal{S}_0$ while a small region accounts for almost all of its failures. This is not an artifact of uneven data, since the policy was trained on episodes whose initial states were spread evenly over $\mathcal{S}_0$, and its failures still cluster (per-region success rates in Sec. G.1). Collection from uniformly sampled initial states then spends most of the operator's time on states the policy already handles. Autonomous RL receives most of its learning signal from states of *intermediate difficulty*, where the policy sometimes succeeds and sometimes fails, and curricula are designed to stay in that regime [41, 50, 63]. A human in the loop removes the upper limit of this regime, since the operator can complete the task from any state, and the resulting correction demonstrates the missing behavior exactly where it is needed. Collection can therefore target the hardest states found so far, with no need to calibrate their difficulty more finely. **DY:** I find the progression between these ideas hard to follow. The intended logic seems to be: clustered failures - uniform collection is inefficient - autonomous RL usually targets intermediate difficulty - HiL lets us also target very hard states. Could we add short transitions to make that progression explicit? Suggested rewrite: "This clustering suggests that uniform collection can waste operator time... In autonomous RL... In our HiL setting, however..." The remaining question is how to keep finding them. Our collection strategy balances returning to the regions where the policy has been observed to fail with maintaining coverage of $\mathcal{S}_0$, so that new failures surface as known ones are fixed. Alg. 2 gives the procedure that selects each round's initial states. We describe its two steps, failure promotion and coverage fill, and then introduce *counterfactual demonstrations*, which complement corrections at the selected initial states.

Algorithm 2 SELECTINITIALSTATES: performance-guided initial-state selection, the Analyze step of Alg. 1

Require: budget M; collected initial states $\mathcal{P}$; initial states $\mathcal{F}$ of the round's evaluation failures; promotion cap m_f; nearest-neighbor distance $d(s, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|s - y\|$

$\mathcal{G} \leftarrow$ up to m_f states from $\mathcal{F}$, earliest failure stage first ▷ failure promotion
while $|\mathcal{G}| < M$ **do**
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{\arg\max_{s \in \mathcal{S}_0} d(s, \mathcal{P} \cup \mathcal{G})\}$ ▷ coverage fill
end while
return $\mathcal{G}$

Failure promotion. An observed failure is the best available evidence of where the policy tends to fail. Each round therefore initializes part of its episodes at the exact initial states where the deployed policy failed in the evaluation just completed. Failures are taken in order of the task stage at which the policy failed, the earliest and thus most severe first (stage labels in Sec. D.1), up to m_f promoted failures, which reserves the rest of the budget for the coverage fill (guardrails in Sec. G.7).

Coverage fill. Coverage begins before any failure has been observed. With a budget of tens of episodes, i.i.d. uniform samples leave gaps and clusters by chance, so the base demonstrations of round 0 are recorded at initial states drawn from a scrambled Sobol sequence [64, 65, 66], a *low-discrepancy* sequence whose points cover $\mathcal{S}_0$ more evenly (comparison in Sec. G.2). In later rounds, the episodes left after promotion keep supervision reaching the parts of $\mathcal{S}_0$ that remain unvisited. Which parts those are depends on everything collected so far, so the coverage fill uses farthest-point selection [67, 68]. Each coverage-fill state is the point of $\mathcal{S}_0$ farthest from every initial state collected so far, this round's promoted failures included, and is therefore the point that improves coverage the most. **DY:** We should define the distance metric here, since "farthest" is otherwise underspecified. For example, if this is normalized Euclidean distance over reset parameters: "FPS repeatedly selects the candidate with the largest minimum normalized Euclidean distance to previously collected initial states."

Counterfactual demonstrations. A correction begins from a state the policy reached on its own and therefore demonstrates recovery from a mistake rather than what the policy should have done. When the policy drops an object in transit, for example, the correction picks it back up, which is useful recovery behavior, but the more important lesson is to carry the object without dropping it. In such cases the operator completes the correction, resets to the same initial state, and demonstrates the whole task from the start, a *counterfactual demonstration* (CF) that places the missing behavior in the actor's support. A correction that only nudges a stuck or slightly misaligned robot the rest of the way already shows the intended behavior, so no CF is recorded. Each CF counts toward the M-episode budget, so rounds with and without CFs cost the same number of episodes. Sec. 4.2 ablates the initial-state selection and the counterfactual demonstrations.

Instantiation. Each round collects $M = 50$ episodes in the real world and $M = 100$ in simulation. The $K = 50$ evaluation episodes of Alg. 1 start from a separate block of initial states, drawn fresh each round and shared by every compared method, so that success is measured on states the policy has not been trained from. Each evaluation block is used once and not reused. In the real world, the operator places objects from the per-episode reset schematic shown in Fig. 2. The per-task selection settings, including an optional bias of the coverage fill toward harder states, are in Secs. G.6 and J.4.

3.2 Policy learning: decoupled actor and critic

The collection procedure above produces both curated human data that is uniformly successful and autonomous rollouts that include outright failures. Our learning setup departs from the algorithms above in what data each model is trained on and in how the critic is trained. The actor π_θ is trained by BC on $\mathcal{D}_h$ alone, whereas prior work folds autonomous successes into the actor's mixture [16, 13]. In our ablation, adding those segments lowers success (Sec. 4.2). The critic (Q_ψ, V_ξ) is instead trained

on all of $\mathcal{D}_{\text{all}}$, so the failures that the actor's data excludes still inform action selection. No gradients pass between the two, so the actor can be any policy class and each component can be evaluated on its own. We make four further modifications to critic training. For the critic, a transition spans the h executed steps of a chunk, so it scores the first h actions of each predicted chunk and bootstraps h steps ahead [62],

$$y = \sum_{i=0}^{h-1} \gamma^i r_{t+i} + \gamma^h (1 - \texttt{done})\, V_\xi(s_{t+h}), \tag{2}$$

where done marks the end of an episode. **AM:** Isn't this inconsistent with eq 1? eq 1 says Q takes in (s, a i), you defined a above as the full chunk length H, not the prefix h executed actions, which is what is implied in this text and eq 2 Over long horizons this one-chunk target propagates reward slowly, so the critic instead regresses onto the λ-return [69], an exponentially weighted average of chunked n-step targets in the spirit of GAE [70],

$$y^\lambda = (1-\lambda) \sum_{n \geq 1} \lambda^{n-1} y^{(n)}, \qquad y^{(n)} = \sum_{i=0}^{nh-1} \gamma^i r_{t+i} + \gamma^{nh}\, V_\xi(s_{t+nh}), \tag{3}$$

truncated at the end of the episode, with λ annealed from near 1 to 0 over the first part of training so that the target reduces to eq. (2) and its bias vanishes (Sec. J.2). Its value head is our reimplementation of distributional implicit value learning (DIVL) [15], which replaces the scalar value with a categorical distribution over returns (Sec. H.2). Finally, because these tasks alternate between large transport motions and small insertion corrections, the critic rescales its action input by a state-conditioned affine transform,

$$\tilde{\mathbf{a}} = \big(\mathbf{a} - \mu(s)\big) \odot \exp\big(-\ell(s)\big), \tag{4}$$

with offsets $\mu(s)$ and log-scales $\ell(s)$ produced by a small network, so that action differences are represented at a scale that depends on the current state (Sec. H.3). At deployment the actor draws N chunks in one batched forward pass and the critic selects among them by eq. (1) (N per task in Sec. H).

4 Experiments

Our experiments address three questions:

Q1. *Comparison.* How does **Mulligan** compare to existing methods? (Sec. 4.1)
Q2. *Ablations.* What does each part of **Mulligan** contribute? (Sec. 4.2)
Q3. *Productivity.* Does learning come at the cost of completing tasks during collection? (Sec. 4.3)

To answer these three questions, we set up our experiments as described below in Sec. 4.0.

4.0 Experiment Setup

Task suite. For real-world experiments, we use the Franka Panda robot platform used in DROID [5] and evaluate 3 tasks: INSERT MARKER (grasp a pen and insert it into a holder), THREAD NUT (grasp a nut and place it onto a peg), and ROUTE CABLE (seating a rope into two clips in succession). All tasks end in a tight-clearance insertion (e.g., $\sim$1 mm for INSERT MARKER) or seating stage where small pose errors cause failure, and objects are randomized over wide initial-state ranges (e.g., 15–46 cm). Fig. 2 and Fig. 1 (left) show the ranges, and Sec. K.3 lists them.

In simulation, we use two Robosuite [71] versions of the same square-nut insertion task. SQUARE-NARROW is the Robomimic [6] Square task: it randomizes the nut pose while keeping the peg fixed. SQUARE-BROAD is the MimicGen [9] variant that expands the nut-position range and also randomizes the peg position (implementation names and versions in Sec. C.3). Thus, "Narrow" and "Broad" name the initial-state distributions, not different insertion goals.

Compared methods. The human-in-the-loop methods vary two design choices. Initial states are either sampled uniformly or chosen by **Mulligan**, which selects them from the policy's observed performance and adds counterfactual demonstrations (Sec. 3.1). The learner is either HG-DAgger [11], which behavior-clones the human data, or HiL-IDQL, which trains the same actor together with an IQL critic on all data and reranks N sampled action chunks at deployment (BoN, Sec. 3.2; N per task in Sec. J.3). We name each method by its learner and append "+**Mulligan**" when collection is performance-guided, giving HG-DAgger, HG-DAgger+**Mulligan**, HiL-IDQL, and HiL-IDQL+**Mulligan**. All four share the actor architecture (a Diffusion Policy U-Net [1]), the critic formulation, and the per-round collection budget, so changing one choice isolates its effect (architectures and hyperparameters in Sec. H). All four run in simulation. In the real world, we run one sequence per collection strategy, so we deploy HG-DAgger against the full method, HiL-IDQL+**Mulligan**. The critic costs no extra collection, since the rollouts it trains on arrive either way. Reranking helps only once on-policy data has accumulated (Sec. 4.2), so the deployed policy is the actor alone in early rounds and the reranked actor after that (teal actor-only points followed by purple reranked points in the HiL-IDQL+**Mulligan** campaign). Each round's data are collected by the policy plotted for the previous round. That campaign's actor evaluated without reranking, HG-DAgger+**Mulligan**, is in Sec. B.1 and omitted from the body figures for clarity. Separately, in simulation we compare against autonomous self-improvement methods that receive no interventions, branching from the shared round-0 checkpoints: self-imitation retrains on its own rollouts [72, 1], filtered BC on only the successful ones [73], and batch-online IDQL trains a critic on all rollouts for BoN reranking [60, 16].

Evaluation metrics. We report autonomous success rate (SR) on held-out initial states and, on SQUARE-NARROW, its complement, failure rate ($100 - \text{SR}$). On ROUTE CABLE, where an episode seats two clips in sequence, the headline figure reports clip success rate, the mean 0–2 clip score divided by two, and full-task success (both clips seated) is reported in the text and appendix. Beyond reliability, we measure execution speed (time-to-completion of successful episodes), throughput (completed task units per robot minute, including time spent by failures), and the rate of human intervention during collection.

Evaluation protocol. Evaluation is separate from collection and runs without interventions. Each round scores every method on one fresh block of 50 Sobol-sampled initial states that no policy has trained from (Sec. 3.1), with the methods interleaved on a shuffled list under labels blind to the operator, and collection is blinded and interleaved the same way. After scoring, the block's rollouts enter $\mathcal{D}_{\text{all}}$, so success rates are comparable within a round but, with a new block each round and possible day-to-day drift, not across rounds. In simulation, every policy trains for a preset number of gradient steps and is scored on one fixed set of initial states over five seeds. Real-world whiskers are Wilson one-standard-error intervals [74] and simulation intervals are seed-level two-sided Student-t 95% confidence intervals. The blinding procedure, the paired per-state tests, and the interval constructions are in Sec. D.

4.1 Q1: How does Mulligan compare to existing methods?

Held-out performance. Fig. 3 shows held-out performance for all five tasks. On all three real tasks, **Mulligan** finishes above uniform sampling under blind, interleaved collection, and pooled over all rounds the improvement is significant on each task (paired tests over initial states, $p < 0.005$ on INSERT MARKER and THREAD NUT, and $p = 0.03$ in full-task success and $p = 0.004$ in clip success rate on ROUTE CABLE). At the final checkpoint, HiL-IDQL+**Mulligan** finishes $+34$, $+17$, and $+14$ percentage points (pp) above HG-DAgger in full-task success on INSERT MARKER, THREAD NUT, and ROUTE CABLE (counts and tests in Sec. B.2). In Fig. 3, ROUTE CABLE is plotted as clip success rate, which ends at 43% for HG-DAgger and 53% for HiL-IDQL+**Mulligan**. The final-checkpoint ROUTE CABLE margin is not significant on its own ($p = 0.14$), while the largest single-checkpoint gap, $+22$ pp at R4, is ($p = 0.027$).

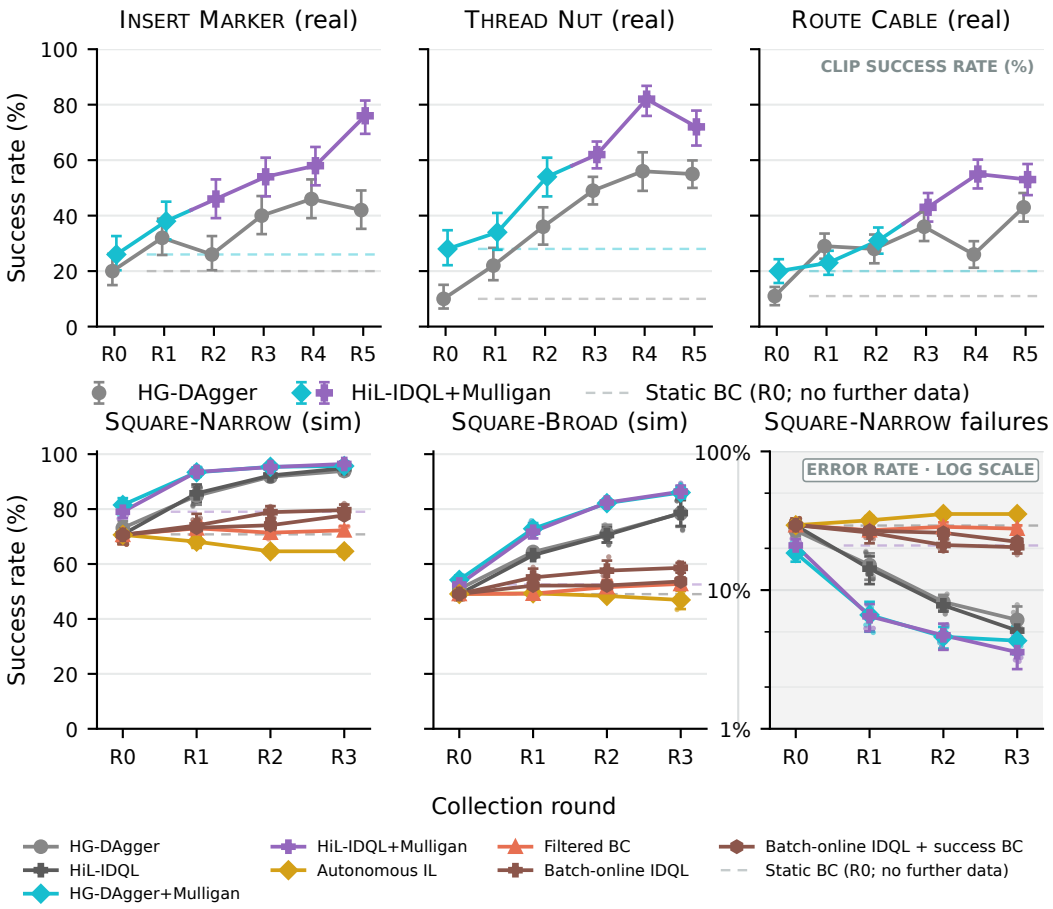

Figure 3: **Headline: held-out performance over DAgger rounds, real world (top) and simulation (bottom). Top:** full-task success rate on INSERT MARKER and THREAD NUT, and clip success rate on ROUTE CABLE: the percentage of the two clip insertions completed per episode, averaged over evaluation episodes. Routing whiskers show standard errors across episodes; the other real panels use Wilson-score one-standard-error intervals. Per-method full-success results for all three tasks are in Sec. B.1. **Bottom:** simulation fixed-grid success rate for SQUARE-NARROW and SQUARE-BROAD, and SQUARE-NARROW failure rate (100 − SR) on a log scale. Compared methods use matched per-round episode budgets. In the real-world HiL-IDQL+**Mulligan** campaign, teal points use the actor alone and purple points use critic reranking. Both simulation success axes start at zero.

In simulation, against prior methods that use no human corrections, the HiL methods show that corrections are one of the most effective levers for improving from on-robot data. Without the human in the loop, only the BoN-reranked methods improve at all, so a value-based objective appears necessary. On SQUARE-NARROW, where success approaches the 100% ceiling, Fig. 3 plots the failure rate on a log scale and shows HiL-IDQL+**Mulligan** continuing to drive failures down through the final round.

Execution speed and throughput. These gains do not come at the expense of execution speed. HG-DAgger has been reported to trade speed for success [13], whereas Fig. 4 shows HiL-IDQL+**Mulligan** also completing more tasks per robot minute than HG-DAgger on all three real tasks (time-to-completion and simulation throughput in Secs. B.3 and B.4).

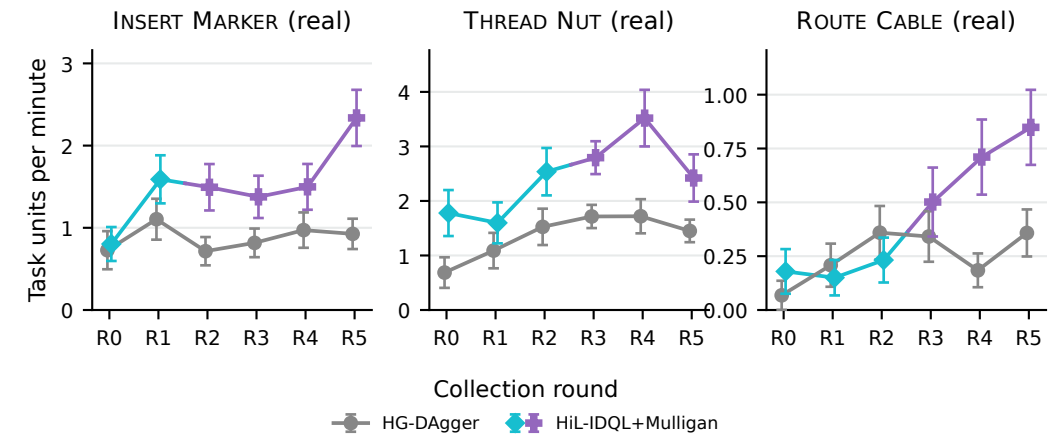

Figure 4: Completed tasks per robot minute on INSERT MARKER, THREAD NUT, and ROUTE CABLE, counting time spent on both successful and failed episodes. A completed ROUTE CABLE task requires both clips seated. HiL-IDQL+**Mulligan** follows the deployed campaign, with actor-only early rounds in teal and critic-reranked rounds in purple; whiskers show bootstrap standard errors. Complete per-method real-world results and simulation results are in Secs. B.3 and B.4.

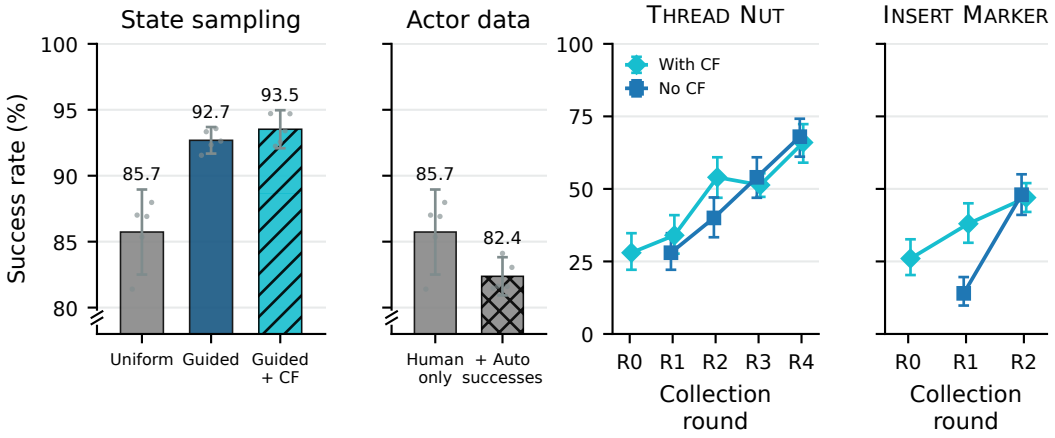

Figure 5: **Component ablations in simulation (left two panels) and the real world (right two). State sampling:** SQUARE-NARROW round-1 grid success for uniform, guided, and guided-with-CF collection. **Actor data:** adding successful autonomous segments lowers mean success relative to human-only training. Simulation bars show five-seed means, seed dots, and 95% Student-t intervals using the frozen DIVL evaluations in Sec. B.5. **Counterfactual demos:** THREAD NUT and INSERT MARKER with and without CF demos, through the last measured no-CF round. Whiskers show Wilson one-standard-error intervals (z=1); R0 is the shared source actor, shown once.

4.2 Q2: What does each component contribute?

Data-collection strategies. *Initial-state sampling strategy:* Holding learning fixed, we compare collection strategies. Low-discrepancy Sobol sampling of the round-0 initial states yields equal or better demo-only point estimates on all five tasks, with the largest gain on THREAD NUT and SQUARE-NARROW and smaller gains on INSERT MARKER, ROUTE CABLE, and SQUARE-BROAD, ranging in simulation from +8.2 pp on SQUARE-NARROW to +3.5 pp on SQUARE-BROAD (round-0 tables in Sec. B.5). In Fig. 5 (first panel), we compare held-out success on SQUARE-NARROW round 1. Mean success increases from uniform collection to performance-guided collection (Sec. 3.1), with a smaller further increase when adding counterfactual demos. The full com-

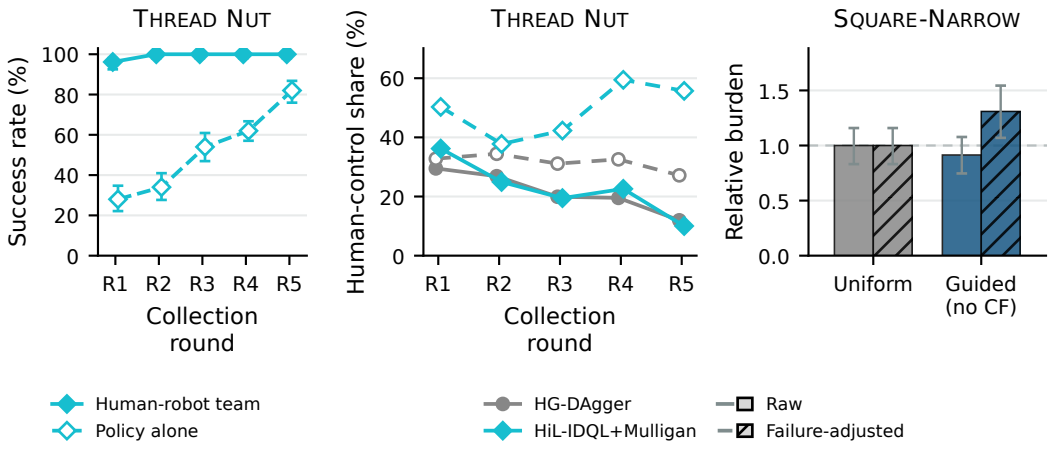

Figure 6: Fresh collection and held-out collector success on THREAD NUT (Wilson $z{=}1$). All three tasks and protocol details are in Sec. B.8.

Figure 7: Human burden: THREAD NUT human-control frame share (left) and SQUARE-NARROW intervention-episode share (right), raw and divided by collector failure rate. Sim shares are indexed to uniform collection; Wilson 95% bounds treat normalization factors as fixed. Real-world curves compare the HG-DAgger and HiL-IDQL+**Mulligan** collection campaigns; simulation bars compare uniform and guided collection without CF demos. Both exclude CF replay. All real tasks and metric definitions are in Sec. B.9.

parison, including Sobol sampling, is in Sec. B.5; sampler definitions and per-round counts are in Sec. G.

Counterfactual demonstrations: Fig. 5 (right two panels) compares performance-guided collection with and without CF demos on THREAD NUT and INSERT MARKER. On INSERT MARKER, round-1 success is 38% with CF versus 14% without; the arms reach 47% and 48% at round 2. The early gap is smaller on THREAD NUT: 34% versus 28% at round 1 and 54% versus 40% at round 2, with similar performance by rounds 3–4. These results suggest that CF demos are most useful early, when corrections from difficult initial states may provide recovery behavior without enough examples of completing the task. The larger THREAD NUT panel and evaluation details are in Sec. B.6.

Decoupled actor- and critic-learning *Value-augmented sampling:* Holding collection fixed, we compare the actor alone with the actor reranked by the critic (success of every method in every round in Sec. B.1). Reranking provides little benefit early and raises success only in later rounds, once enough on-policy data is collected, so performance-guided collection accounts for most of the early gains and the two contribute at different times and in different parts of the task. Decomposing INSERT MARKER and THREAD NUT into grasp and insertion stages (stage-wise success in Sec. B.7), performance-guided collection leads uniform sampling at the grasp stage in early rounds and reaches parity once both methods have collected enough data, while the value function's gains concentrate at the precise insertion stage.

Actor-critic data curation: In Fig. 5 (second panel), adding successful autonomous segments to the actor's human data lowers mean success by 3.4 percentage points, with a 95% Welch interval of $[-6.5, -0.2]$. Full estimates and critic/data-mixture ablations are in Secs. B.5 and I.

4.3 Q3: Does learning impede productivity?

Deployment-time collection is productive. A human operator supervises every DAgger collection episode and can take over when the policy fails. Despite low initial success rates and performance-guided sampling, the human-robot team completes 95–100% of episodes per round on all three real tasks from the first round, even though a mistake the operator cannot recover from,

such as an object knocked off the table, ends the episode as a failure. Fig. 6 shows this for THREAD NUT, and Sec. B.8 for all three tasks. That is, learning and productive deployment appear complementary, as each round is a batch of tasks completed with near-perfect reliability that also effectively improves policy performance. Additionally, the operator's intervention burden shrinks as learning progresses.

Burden relative to collector performance. In Fig. 7, the raw share of human-controlled frames (solid) is similar for both methods and generally declines across rounds, despite their different held-out policy success rates. Dividing this share by the collector's held-out failure rate $(1 - \mathrm{SR})$ gives a higher ratio for **Mulligan** on THREAD NUT (dashed). The simulation comparison likewise changes from lower raw intervention incidence to higher failure-adjusted incidence under guided sampling. This is consistent with directing supervision toward difficult states, but the ratio also depends on intervention duration and the held-out failure-rate divisor; it does not directly measure the sampled states' failure probability.

5 Discussion

We studied supervised deployment, in which an operator places the objects for each episode and intervenes when the policy fails, and asked how to get the most improvement from each supervised episode. Choosing each round's initial states from the policy's observed failures, and recording counterfactual demonstrations where a correction would only teach recovery, improves the policy faster than uniform collection at the same budget on three real and two simulated tasks. A critic trained on the rollouts that arrive anyway, failures included, adds further gains in later rounds by reranking the actor's samples, and the two gains combine for the two learners we tested.

Several limitations point to future work. First, **Mulligan** assumes initial states can be parameterized and placed at collection time, which supervised collection provides but a fully autonomous deployment may not. There, the robot could only select among naturally arriving initial states, or drive itself to informative ones. Second, we evaluated **Mulligan** with one imitation learner and one value-based learner under one formulation of the human-in-the-loop MDP, e.g., bootstrapping value targets across intervention boundaries (Sec. 3.2). How the same collection strategy performs with other base RL methods and other base policies, including generalist policies updated between rounds, is open. Third, the critic's gains stay outside the actor, since retraining on autonomous rollouts degrades it (Sec. 4.2), and distilling the reranked behavior back into the actor could compound gains across rounds. More broadly, our results suggest that improving a deployed policy is a question not only of how to learn from experience but also of which experience to collect.

Acknowledgments

We thank Andy for help with running evaluations, Ajay for help setting up the physical robot, and Colin for helpful discussions early in the project. We also thank Marcel and Anthony for feedback on the draft. We thank Yotta Labs for providing compute resources. **[TODO:** full names; add funding/grant acknowledgments if applicable**]**

References

[1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An Open-Source Vision-Language-Action Model, June 2024. URL http://arxiv.org/abs/2406.09246. arXiv:2406.09246 [cs].

[5] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, C. Finn, and D. Sadigh. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

[6] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kacholia, Y. Zhu, L. Fan, A. Garg, S. Savarese, L. Fei-Fei, R. Martín-Martín, and F. Xia. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[7] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. ALOHA Unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

[8] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement – residual RL for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[9] A. Mandlekar, Y. Zhu, L. Fan, D. Xu, R. Martín-Martín, L. Fei-Fei, S. Savarese, A. Garg, and F. Xia. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

[10] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[11] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[12] P. Wu, Y. Shentu, Q. Liao, D. Jin, M. Guo, K. Sreenath, X. Lin, and P. Abbeel. RoboCopilot: Human-in-the-loop interactive imitation learning for robot manipulation. *arXiv preprint arXiv:2503.07771*, 2025.

[13] Z. Hu, R. Wu, N. Enock, J. Li, R. Kadakia, Z. Erickson, and A. Kumar. RaC: Robot learning for long-horizon tasks by scaling recovery and correction. *arXiv preprint arXiv:2509.07953*, 2025.

[14] Physical Intelligence. $\pi_0.6$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[15] Y. Wang, X. Li, P. Xie, P. Yang, B. Nie, Y. Cai, Q. Zhang, C. Qu, J. Wu, J. Song, X. Ren, J. Huang, M. Pan, S. Feng, Z. Chen, and J. Luo. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies, 2026. URL https://arxiv.org/abs/2605.00416.

[16] P. Dong, S. Mirchandani, D. Sadigh, and C. Finn. What matters for batch online reinforcement learning in robotics? *arXiv preprint arXiv:2505.08078*, 2025.

[17] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. SERL: A software suite for sample-efficient robotic reinforcement learning. *arXiv preprint arXiv:2401.16013*, 2024.

[18] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *arXiv preprint arXiv:2410.21845*, 2024.

[19] Y. Chen, S. Tian, S. Liu, Y. Zhou, H. Li, and D. Zhao. ConRFT: A reinforced fine-tuning method for VLA models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025.

[20] J. Luo, F. Dong, Y. Zhai, Y. Ma, and S. Levine. RLIF: Interactive imitation learning as reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[21] P. Dong, K.-H. Hung, T. Gao, D. Sadigh, and C. Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models, 2026. URL https://arxiv.org/abs/2605.25477.

[22] K. Lei, H. Li, D. Yu, Z. Wei, L. Guo, Z. Jiang, Z. Wang, S. Liang, and H. Xu. RL-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint arXiv:2510.14830*, 2025.

[23] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi. Residual off-policy RL for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025.

[24] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, F. Hu, J. Wu, Z. Luo, L. Fan, G. Shi, and Y. Zhu. Self-improving vision-language-action models with data generation via residual RL. *arXiv preprint arXiv:2511.00091*, 2025.

[25] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[26] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg. Fleet-DAgger: Interactive robot fleet learning with scalable human supervision. In *Conference on Robot Learning (CoRL)*, 2022.

[27] J. Spencer, S. Choudhury, A. Venkatraman, B. Zhan, and J. A. Bagnell. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *Robotics: Science and Systems (RSS)*, 2020.

[28] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023.

[29] P. Dong, Q. Li, D. Sadigh, and C. Finn. EXPO: Stable reinforcement learning with expressive policies. *arXiv preprint arXiv:2507.07986*, 2025.

[30] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[31] J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 21, 2009.

[32] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

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

[34] Y. Seo, J. Uruç, and S. James. Continuous control with coarse-to-fine reinforcement learning. *arXiv preprint arXiv:2407.07787*, 2024.

[35] H. Hu, S. Mirchandani, and D. Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

[36] 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. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7772–7779, 2024.

[37] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. *arXiv preprint arXiv:2310.15145*, 2023.

[38] S. K. S. Ghasemipour, A. Wahid, J. Tompson, P. Sanketi, and I. Mordatch. Self-improving embodied foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[39] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2002.

[40] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2017.

[41] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning (ICML)*, 2018.

[42] T. Salimans and R. Chen. Learning Montezuma's Revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[43] C. Resnick, R. Raileanu, S. Kapoor, A. Peysakhovich, K. Cho, and J. Bruna. Backplay: "Man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

[44] S. Dai, A. Hofmann, and B. Williams. Automatic curricula via expert demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[45] S. Tao, A. Shukla, T. kai Chan, and H. Su. Reverse forward curriculum learning for extreme sample and demonstration efficiency in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[46] M. Bauza, J. E. Chen, V. Dalibard, N. Gileadi, R. Hafner, M. F. Martins, J. Moore, R. Pevceviciute, A. Laurens, D. Rao, M. Zambelli, M. Riedmiller, J. Scholz, K. Bousmalis, F. Nori, and N. Heess. DemoStart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. *arXiv preprint arXiv:2409.06613*, 2024.

[47] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, S. Levine, and K. Hausman. Jump-start reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

[48] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return, then explore. *Nature*, 590:580–586, 2021.

[49] M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning (ICML)*, 2021.

[50] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[51] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[52] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[53] J. Gao, A. Xie, T. Xiao, C. Finn, and D. Sadigh. Efficient data collection for robotic manipulation via compositional generalization. In *Robotics: Science and Systems (RSS)*, 2024.

[54] S. Sagar, J. Duan, S. Vasudevan, Y. Zhou, H. B. Amor, D. Fox, and R. Senanayake. RoboMD: Uncovering robot vulnerabilities through semantic potential fields. *arXiv preprint arXiv:2412.02818*, 2024.

[55] U. B. Karli and T. Fitzgerald. RECALL: Recovery experience collection for active lifelong learning in vision-language-action models. *arXiv preprint arXiv:2606.23617*, 2026.

[56] G. Zhao, Z. Tang, X. Chen, Z. Kuang, Y. Tian, and G. Li. FLARE: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[57] H. Hao, S. N. Syed, J. Ichnowski, and J. Schneider. FAR: Failure-aware retry for test-time recovery and continual policy improvement. *arXiv preprint arXiv:2607.01111*, 2026.

[58] S. Huang, Y. Liao, S. Feng, S. Jiang, S. Liu, H. Li, M. Yao, and G. Ren. Adversarial data collection: Human-collaborative perturbations for efficient and robust robotic imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[59] M. Hou, K. Hindriks, A. E. Eiben, and K. Baraka. Active robot curriculum learning from online human demonstrations. *arXiv preprint arXiv:2503.02277*, 2025.

[60] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. IDQL: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[61] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[62] Q. Li, Z. Zhou, and S. Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[63] L. Bailey, K. Wen, K. Dong, T. Hashimoto, and T. Ma. Scaling self-play with self-guidance. *arXiv preprint arXiv:2604.20209*, 2026.

[64] I. M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. Russian original: Zh. Vychisl. Mat. i Mat. Phys., 7:784–802.

[65] A. B. Owen. Scrambling Sobol and Niederreiter-Xing points. *Journal of Complexity*, 14(4): 466–489, 1998.

[66] S. Joe and F. Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[67] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[68] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[69] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

[70] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

[71] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2020. URL https://arxiv.org/abs/2009.12293.

[72] D. A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1988.

[73] S. Mirchandani, S. Belkhale, J. Hejna, E. Choi, M. S. Islam, and D. Sadigh. So you think you can scale up autonomous robot data collection? In *Conference on Robot Learning (CoRL)*, 2024.

[74] E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[75] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation, 2024. URL https://arxiv.org/abs/2409.09491.

[76] A. Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society*, 54(3):426–482, 1943.

[77] L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001.

[78] Q. McNemar. Note on the sampling error of the difference between correlated proportions. *Psychometrika*, 12(2):153–157, 1947.

[79] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979.

[80] X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model, 2021. URL https://arxiv.org/abs/2101.05982.

[81] P. Dong, K.-H. Hung, A. Swerdlow, D. Sadigh, and C. Finn. Tql: Scaling q-functions with transformers by preventing attention collapse, 2026. URL https://arxiv.org/abs/2602.01439.

[82] C. Parés-Morlans, N. Kuhn, I. Liu, A. Longhini, and J. Bohg. Demystifying when and why vlas fail in contact-rich tasks and how to fix them, 2026. URL https://arxiv.org/abs/2608.01402.

[83] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1709.07871.

[84] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[85] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[86] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[87] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

[88] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2011.

[89] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, 2017.

[90] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4–5):421–436, 2018.

[91] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. MT-Opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[92] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[93] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3651–3657, 2019.

[94] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning (CoRL)*, pages 1101–1112, 2020.

[95] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. The ingredients of real-world robotic reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

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

[97] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[98] Z. Hu, A. Rovinsky, J. Luo, V. Kumar, A. Gupta, and S. Levine. REBOOT: Reuse data for bootstrapping efficient real-world dexterous manipulation. In *Conference on Robot Learning (CoRL)*, 2023.

[99] A. Sharma, A. M. Ahmed, R. Ahmad, and C. Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 3292–3308, 2023.

[100] R. Mendonca, E. Panov, B. Bucher, J. Wang, and D. Pathak. Continuously improving mobile manipulation with autonomous real-world RL. In *Conference on Robot Learning (CoRL)*, 2024.

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

[102] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu, S. Lin, and J. Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

[103] Q. Li and S. Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

[104] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[105] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

[106] S. Park, Q. Li, and S. Levine. Flow q-learning, 2025. URL https://arxiv.org/abs/2502.02538.

[107] X. Yuan, T. Mu, S. Tao, Y. Fang, M. Zhang, and H. Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[108] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[109] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.

[110] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

[111] M. S. Mark, T. Gao, G. G. Sampaio, M. K. Srirama, A. Sharma, C. Finn, and A. Kumar. Policy agnostic RL: Offline RL and online RL fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.