

CUPID: Curating Data your Robot Loves with Influence Functions

Anonymous Author(s)

Affiliation

Address

email

Abstract: In robot imitation learning, policy performance is tightly coupled with the quality and composition of the demonstration data. Yet, developing a precise understanding of how individual demonstrations contribute to downstream outcomes—such as closed-loop task success or failure—remains a persistent challenge. Inspired by the theory of influence functions, we propose CUPID. Given a set of evaluation rollouts, CUPID estimates the influence of a training demonstration on the policy’s expected return. This enables ranking and selection of demonstrations according to their impact on the policy’s closed-loop performance. We use our estimator to curate data by 1) filtering out training demonstrations that harmed the policy’s performance and 2) subselecting newly collected trajectories that will most help improve the policy. Extensive simulated and hardware experiments show that our approach consistently identifies which data drives test-time performance. For example, training with less than 33% of curated data can result in state-of-the-art diffusion policies on the simulated Robomimic benchmark, and we observe similar improvements in hardware experiments. Furthermore, our hardware experiments show that our influence-based estimator can identify robust strategies under distribution shift, isolate spurious correlations, and even enhance post-training of generalist policies.

Keywords: Imitation Learning, Data Curation, Influence Functions

1 Introduction

While some of the largest breakthroughs in deep learning have emerged from architectural innovations, data often remains an underrecognized yet crucial component of a model’s overall performance. In particular, the success of scaling vision and language models has been trailed by a rising interest in data attribution [1, 2, 3]—methods that causally link model behavior to training data—and automatic data curation algorithms [4, 5, 6], grounded in the idea that not all data points contribute equally, or even positively, to a model’s performance. As parts of the robotics community scale imitation learning and robotics datasets become increasingly diverse [7, 8], developing a deeper understanding of (i) how demonstration data shapes policy behavior and (ii) how we can extract maximum utility from training datasets will be imperative to advancing policy performance toward reliable, open-world deployment.

Curating data for robot imitation learning has been the focus of several recent works [9, 10, 11]. A common approach retains demonstrations deemed most valuable under a heuristic task-agnostic *quality* metric, resulting in a smaller dataset curated offline [10]. This approach typically relies on the implicit assumption that the designed quality metric correlates well with the policy’s downstream performance—an alignment that may not hold uniformly across diverse robotics tasks. Although recent efforts attempt to learn curation heuristics correlated with *performance* using on-policy experience [11], they do not establish strong causal links between training data and observed policy behavior, which may result in misidentifying the root cause of success or failure with respect to the training data [12].

In this work, we formally define data curation in robot imitation learning as the problem of identifying which expert demonstrations maximally contribute to the policy’s expected return. We then introduce CUPID (CURating Performance-Influencing Demonstrations), a method that directly targets this objective by leveraging influence functions [13, 14]—a technique popularized in the data attribution literature [15]—to identify which demonstrations most influenced a policy’s predictions during closed-

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.

loop execution. In particular, we show that a demonstration’s influence on the policy’s expected return decomposes into a tractable sum over state-action transitions in the demonstration, and can be efficiently approximated over a set of policy rollouts using a REINFORCE-style estimator [16]. Ranking demonstrations by their estimated performance impact facilitates curation in two settings: (a) filtering existing demonstrations from training sets and (b) selecting high-impact demonstrations from newly collected data—whereas prior work focuses solely on filtering [10, 11]. Finally, while our approach offers a general and effective standalone signal for curating demonstrations, we investigate how its utility is affected under combined use with task-agnostic quality metrics (also derived from influence scores), identifying conditions where their integration strengthens or weakens overall curation performance.

Our contributions are three-fold: (1) We formulate robot data curation as the problem of valuating demonstrations in accordance with their downstream impact on policy performance; (2) We propose CUPID, a novel approach for curating imitation learning datasets based on influence functions, causally linking demonstrations to the policy’s expected return; (3) We characterize the conditions under which the integration of task-agnostic quality metrics strengthens performance-based data curation, and provide practical insights on when such integration is beneficial. Extensive simulation and hardware experiments demonstrate that data curation with CUPID significantly improves policy performance in mixed-quality regimes with a fraction of the training data. Moreover, it successfully identifies robust strategies under test-time distribution shifts, and can even disentangle spurious correlations in training data that hinder generalization, all by observing only policy outcomes without requiring additional supervision.

2 Related Work

Data Curation in Robotics. While assembling larger datasets has become central to scaling efforts in imitation learning [17, 18, 8, 7, 19], policy performance also depends on data composition [20, 21, 22, 23]. Thus, several works extract greater utility from robotics datasets via data augmentation [24, 25, 26, 27, 28] and mixture optimization [29]. Only recently has attention shifted to valuating individual demonstrations for data curation [9, 10, 11]. Hejna et al. [10] estimate demonstration quality offline via mutual information—without considering policy performance. Closest to our work is Demo-SCORE [11], which trains classifiers to distinguish successful and failed rollouts across multiple policy checkpoints. In contrast, we measure the causal influence of each demonstration on the policy’s return, providing a signal that (a) does not require observing both successes and failures, (b) uses only a single policy checkpoint, and (c) is more robust to spurious correlations in the rollout distribution. Further, our approach naturally extends to selecting new data, whereas [10, 11] exclusively focus on filtering existing training sets.

Data Attribution outside Robotics. Data attribution methods model the relationship between training data and model behavior, with applications in model interpretability [30, 2], data valuation [31, 32, 33], and machine unlearning [34]. Much of recent work focuses on improving the reliability of attribution algorithms [35, 36], such as influence functions [13, 14], and extending them to increasingly complex generative architectures [1, 37, 38]. Engstrom et al. [3] show that the performance of large language models can be improved by training on datasets curated through data attribution. However, their setting assumes identical training and evaluation objectives (i.e., prediction loss) and access to test-time labels. In contrast, robot imitation learning involves an objective mismatch: policies are trained via supervised learning but evaluated through closed-loop environment interactions, where success depends on many sequential predictions and ground-truth action labels are unavailable at test-time.

3 Background: Data Attribution via Influence Functions

The goal of data attribution methodologies is to explicitly relate model performance and behavior to the training data \mathcal{D} , so that we can answer *counterfactual* questions about the contribution of training samples towards test-time predictions. Consider a standard supervised learning setting, where we fit model parameters θ on a given training dataset $\mathcal{D} := \{z^1, \dots, z^n\}$ with $\theta(\mathcal{D}) = \arg \min_{\theta'} \{R(\theta'; \mathcal{D}) := \frac{1}{n} \sum_{i=1}^n \mathcal{L}(z^i; \theta')\}$. Moreover, let $f(\hat{z}; \theta) \in \mathbb{R}$ be any chosen performance metric on a test sample \hat{z} given model parameters θ (e.g., cross-entropy loss for a classifier). Then, a data attribution method $\Psi^{\text{out}} : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}$ aims to approximate the change in the

performance metric f if we were to exclude sample z^i from the model’s training data. That is, we aim to design Ψ^{out} such that $\Psi^{\text{out}}(\hat{z}, z^i) \approx f(\hat{z}; \theta(\mathcal{D} \setminus z^i)) - f(\hat{z}; \theta(\mathcal{D}))$.

The influence function is a data attribution technique that approximates Ψ^{out} *without* retraining any models [15]. Consider perturbing the training objective as $R_{\epsilon, z}(\theta'; \mathcal{D}) := R(\theta'; \mathcal{D}) + \epsilon \mathcal{L}(z, \theta')$, where we add an infinitesimal weight ϵ on some sample z to R . The *influence function* estimates the change in the performance metric f as a function of ϵ with a first-order Taylor approximation as

$$\Psi_{\text{inf}}(\hat{z}, z) := \left. \frac{df(\hat{z}; \theta)}{d\epsilon} \right|_{\epsilon=0} = -\nabla_{\theta} f(\hat{z}; \theta(\mathcal{D}))^{\top} H_{\theta}^{-1} \nabla_{\theta} \mathcal{L}(z; \theta(\mathcal{D})), \quad (1)$$

where $H_{\theta} = \frac{1}{n} \sum_{i=1}^n \nabla_{\theta}^2 \mathcal{L}(z^i; \theta(\mathcal{D}))$ denotes the Hessian of the training loss¹ [13]. Therefore, we can use the influence function to directly approximate the *leave-one-out* influence Ψ^{out} of a sample $z^i \in \mathcal{D}$ as $\Psi_{\text{inf}}^{\text{out}}(\hat{z}, z^i) := -\frac{1}{n} \Psi_{\text{inf}}(\hat{z}, z^i)$. In addition, for $z \notin \mathcal{D}$ we similarly define the *add-one-in* influence as $\Psi_{\text{inf}}^{\text{in}}(\hat{z}, z) := \frac{1}{n} \Psi_{\text{inf}}(\hat{z}, z) \approx f(\hat{z}; \theta(\mathcal{D} \cup \{z\})) - f(\hat{z}; \theta(\mathcal{D}))$.

4 Problem Formulation

Imitation Learning (IL): The objective of this work is to understand how demonstration data contributes to closed-loop performance in robot imitation learning. Thus, we consider a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \rho_0 \rangle$ with state space \mathcal{S} , action space \mathcal{A} , transition model \mathcal{T} , reward model R , initial state distribution ρ_0 , and finite horizon H . We train a policy π_{θ} to minimize a behavioral cloning (BC) objective, i.e., $\theta = \arg\min_{\theta'} \{\mathcal{L}_{\text{bc}}(\theta'; \mathcal{D}) := \frac{1}{|\mathcal{D}|H} \sum_{\xi^i \in \mathcal{D}} \sum_{(s,a) \in \xi^i} \ell(s, a; \pi_{\theta'})\}$, using a dataset of n expert demonstrations $\mathcal{D} = \{\xi^1, \dots, \xi^n\}$. Each demonstration $\xi^i = ((s_0^i, a_0^i), \dots, (s_H^i, a_H^i))$ consists of a state-action trajectory where the robot successfully completes the task. We treat a trajectory $\tau = (s_0, a_0, \dots, s_H)$ as either a *success* or a *failure*, corresponding to the binary returns $R(\tau) = 1$ and $R(\tau) = -1$ respectively.

Therefore, in IL, we train the policy π_{θ} to match the distribution of successful behaviors in \mathcal{D} , rather than directly maximizing its expected return $J(\pi_{\theta}) := \mathbb{E}_{p(\tau|\pi_{\theta})}[R(\tau)]$. As a result, the policy’s performance is intimately linked to the relative suboptimality—that is, the *quality*—of the demonstration data, not just validation losses, model capacity, or bias-variance tradeoffs. This makes it extremely challenging to systematically improve performance: Recent efforts underscore that simply scaling demonstration collection may result in datasets that contain substantial redundancies and behaviors that may actually harm policy performance, even though $R(\xi^i) = 1$ for all demonstrations $\xi^i \in \mathcal{D}$ [39].

Robot Data Curation: While some recent works propose intuitive measures of data quality to curate data, we find that such heuristics can misalign with how deep models actually learn, sometimes even worsening test-time performance compared to randomly choosing samples (see §6). Therefore, we first formally define robot data curation as the problem of identifying demonstration data that maximizes the policy’s closed-loop performance. In particular, assume that we have a *base policy* π_{θ} trained on the demonstration data \mathcal{D} . We consider two settings that are essential to a policy debugging toolchain. The first is that of *data filtering*, where our goal is to identify and remove redundant or harmful demonstrations from \mathcal{D} that may be hurting the performance of the base policy π_{θ} .

Task 1 (Filter- k demonstrations). Let $\Xi_k^- = \{S \subseteq \mathcal{D} \mid |S| = k\}$ denote all possible k -demonstration subsets of the training dataset $\mathcal{D} = \{\xi^1, \dots, \xi^n\}$, where $k \leq n$. Determine which k demonstrations should be removed from \mathcal{D} to maximize policy performance with respect to the task objective J . That is, find

$$S^* = \arg \max_{S \in \Xi_k^-} J(\pi_{\theta}) \quad \text{s.t.} \quad \theta = \arg \min_{\theta'} \mathcal{L}_{\text{bc}}(\theta'; \mathcal{D} \setminus S).$$

The second is that of *data selection*, where we seek to guide the subselection of new data to maximally improve our base policy, given a fixed budget.

Task 2 (Select- k demonstrations). Let $\Xi_k^+ = \{S \subseteq \mathcal{H} \mid |S| = k\}$ denote all possible k -demonstration subsets of a holdout demonstration dataset $\mathcal{H} = \{\xi^1, \dots, \xi^{n'}\}$, where $k \leq n'$. Determine which k

¹ To reduce the computational cost of Eq. 1, we use TRAK [2], which leverages random projections and an efficient Gauss-Newton Hessian approximation. This also makes the influence function amenable to the non-smooth, non-convex loss functions in practical deep learning problems, so we assume Eq. 1 is well-defined throughout this paper.

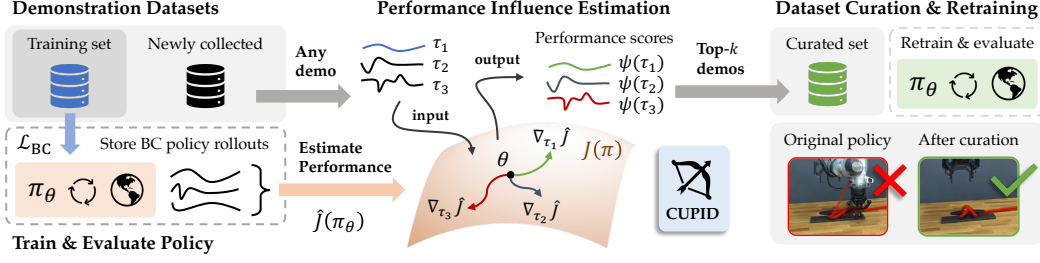


Figure 1: **Overview of curation with CUPID.** Upon training a policy on a set of demonstrations, we evaluate it online to collect closed-loop rollout trajectories, which are used to estimate the policy’s expected return. CUPID ranks demonstration based on their measured influence on this performance estimate and selects the top- k . Thus, curating with CUPID results in a dataset of demonstrations that most strongly influences closed-loop policy success.

132 *demonstrations should be added to \mathcal{D} from \mathcal{H} to maximize policy performance with respect to the task*
 133 *objective J . That is, find*

$$S^* = \arg \max_{S \in \Xi_k^+} J(\pi_\theta) \quad \text{s.t.} \quad \theta = \arg \min_{\theta'} \mathcal{L}_{bc}(\theta'; \mathcal{D} \cup S).$$

134 In **Task 2**, we consider the problem of identifying the most impactful trajectories from a newly collected
 135 batch of demonstrations or from an existing pre-collected dataset, akin to performing quality control.

136 **Policy Testing & Evaluation:** To make progress on **Task 1** and **Task 2**, we assume access to a small
 137 dataset of m rollouts $\mathcal{D}_\tau = \{\tau^1, \dots, \tau^m\} \stackrel{\text{iid}}{\sim} p(\tau | \pi_\theta)$ of the base policy π_θ along with success/failure labels
 138 $\{R(\tau^1), \dots, R(\tau^m)\}$ to estimate $J(\pi_\theta)$. This aligns with how we currently evaluate policies in prac-
 139 tice [40], despite lacking principled strategies to leverage evaluations towards BC policy improvement.

140 5 CUPID: Curating Performance-Influencing Demonstrations

141 While recent works value demonstration data upon heuristic notions of quality [10, 11, 41], **our key in-**
 142 **sight** is that solving curation problems, i.e., **Task 1** and **Task 2** (§4), requires causally connecting training
 143 data to the policy’s closed-loop performance. Therefore, we first adapt techniques from data attribution,
 144 as defined in §3, to directly compute the influence of a training demo on the performance of a policy.
 145 This allows us to use our *performance influence* to directly curate data in alignment with our objectives.

146 5.1 Demonstration-Performance Influence

147 Because the BC training objective is not always reflective of a policy’s closed-loop performance [42], we
 148 must first develop an analogous notion of the *influence function* to capture the impact of a *demonstration*
 149 *trajectory* on the *closed-loop performance* of an imitation learned policy. To do so, we first group the
 150 BC training objective into trajectory-level losses by introducing $\mathcal{L}_{\text{traj}}(\xi; \pi_\theta) := \frac{1}{H} \sum_{(s,a) \in \xi} \ell(s,a; \pi_\theta)$,
 151 so that $\mathcal{L}_{bc}(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\xi^i \in \mathcal{D}} \mathcal{L}_{\text{traj}}(\xi^i; \pi_\theta)$. We now formally define the *performance influence* of
 152 a demonstration as the application of the influence function (see Eq. 1) on the policy’s expected return:

153 **Definition 1** (Performance Influence). *Let ξ be a demonstration of interest. Suppose we train a*
 154 *policy π_θ to minimize the perturbed BC objective $\mathcal{L}_{bc}^{\epsilon, \xi}(\theta; \mathcal{D}) := \mathcal{L}_{bc}(\theta; \mathcal{D}) + \epsilon \mathcal{L}_{\text{traj}}(\xi; \pi_\theta)$. Then,*
 155 *demonstration ξ ’s **performance influence** is the derivative of the policy’s expected return $J(\pi_\theta)$ with*
 156 *respect to the weight ϵ . That is,*

$$\Psi_{\pi\text{-inf}}(\xi) := \left. \frac{dJ(\pi_\theta)}{d\epsilon} \right|_{\epsilon=0} = -\nabla_\theta J(\pi_\theta) H_{bc}^{-1} \nabla_\theta \mathcal{L}_{\text{traj}}(\xi; \pi_\theta),$$

157 where $H_{bc} := \nabla_\theta^2 \mathcal{L}_{bc}(\theta; \mathcal{D})$ denotes the Hessian of the BC objective.

158 In essence, **Definition 1** allows us to answer the counterfactual question “how would the policy’s
 159 expected return change if we upweighted—or by negating, downweighted—a demonstration ξ during
 160 training?” While **Definition 1** neatly aligns with the standard definition of the influence function
 161 in Eq. 1 using J as the performance metric and $\mathcal{L}_{\text{traj}}$ as the demonstration-level loss function, we

differentiate the *performance influence* from the standard influence function based on two key reasons: (1) The performance influence attributes the *outcome* of a policy’s sequential decisions to time-series demonstrations, whereas the existing techniques discussed in §3 only relate an individual labeled prediction to a single training sample; (2) We cannot directly compute $\Psi_{\pi\text{-inf}}$ because the policy’s expected return $J(\pi_\theta)$ depends on the unknown transition dynamics and reward function. To alleviate these challenges, we show that we can decompose the *performance influence* into influence scores of individual action predictions, which we define as the *action influence*.

Definition 2 (Action Influence). *The action influence of a state-action pair (s, a) on a test state-action pair (s', a') is the influence of (s, a) on the policy’s log-likelihood $\log \pi_\theta(a' | s')$. That is,*

$$\Psi_{a\text{-inf}}((s', a'), (s, a)) := -\nabla_\theta \log \pi_\theta(a' | s') H_{bc}^{-1} \nabla_\theta \ell(s, a; \pi_\theta). \quad (2)$$

The advantage of the *action influence* is that we can easily compute the quantities in Eq. 2 given the policy weights θ and the training demos \mathcal{D} , e.g., using the attribution methods discussed in §3. However, we emphasize that computing *action influences* over samples from a policy rollout $\tau \sim p(\tau | \pi_\theta)$ only tells us what demonstration data led to the policy taking those actions, without ascribing value to the resulting outcome (e.g., success or failure). We now show that the performance influence decomposes into the sum of individual action influences, weighted by the trajectory return $R(\tau)$.

Proposition 1. *Assume that $\theta(\mathcal{D}) = \arg \min_{\theta'} \mathcal{L}_{bc}(\theta'; \mathcal{D})$, that \mathcal{L}_{bc} is twice differentiable in θ , and that $H_{bc} \succ 0$ is positive definite (i.e., $\theta(\mathcal{D})$ is not a saddle point)¹. Then, it holds that²*

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau | \pi_\theta)} \left[\frac{R(\tau)}{H} \sum_{(s', a') \in \tau} \sum_{(s, a) \in \xi} \Psi_{a\text{-inf}}((s', a'), (s, a)) \right]. \quad (3)$$

In brief, we prove Proposition 1 using the log-derivative trick underlying policy gradient methods [16, 43] to decompose $\Psi_{\pi\text{-inf}}$ into $\Psi_{a\text{-inf}}$ (see the Appendix). Because Proposition 1 relates the performance influence to the average action influence that a demonstration ξ has on the closed-loop distribution of policy rollouts, Proposition 1 directly provides a method to estimate $\Psi_{\pi\text{-inf}}$:

Estimate $\Psi_{\pi\text{-inf}}$: First, evaluate the policy π_θ by gathering a set of rollouts $\mathcal{D}_\tau = \{\tau^1, \dots, \tau^m\} \stackrel{\text{iid}}{\sim} p(\tau | \pi_\theta)$ and their associated returns $\{R(\tau^1), \dots, R(\tau^m)\}$. Then, construct an empirical estimate of the performance influence $\hat{\Psi}_{\pi\text{-inf}}$ using Eq. 3, by averaging action influences across the rollouts in \mathcal{D}_τ .

5.2 Data Curation with Performance Influence

In this section, we leverage the performance influence $\Psi_{\pi\text{-inf}}$, which we developed in §5.1, to curate data towards the filtering and selection tasks (Task 1 and Task 2) defined in §4. In particular, we use the estimates of $\Psi_{\pi\text{-inf}}$ to make the following first-order Taylor approximations on the *leave-one-out* and *add-one-in* influence (as defined in §3) of a demonstration trajectory as

$$\Psi_{\pi\text{-inf}}^{\text{out}}(\xi) := -\frac{\hat{\Psi}_{\pi\text{-inf}}(\xi)}{|\mathcal{D}|} \approx J(\pi_{\theta(\mathcal{D}/\xi)}) - J(\pi_{\theta(\mathcal{D})}), \quad \Psi_{\pi\text{-inf}}^{\text{in}}(\xi) := \frac{\hat{\Psi}_{\pi\text{-inf}}(\xi)}{|\mathcal{D}|} \approx J(\pi_{\theta(\mathcal{D} \cup \{\xi\})}) - J(\pi_{\theta(\mathcal{D})}).$$

Then, we use the *leave-one-out* and *add-one-in* influences to counterfactually estimate the change in expected return when removing or adding a set of demonstrations S with a linear approximation as $\widehat{\Delta J}(\pi_{\theta(\mathcal{D}/S)}) \propto \frac{1}{|S|} \sum_{\xi \in S} \hat{\Psi}_{\pi\text{-inf}}^{\text{out}}(\xi)$ and $\widehat{\Delta J}(\pi_{\theta(\mathcal{D} \cup S)}) \propto \frac{1}{|S|} \sum_{\xi \in S} \hat{\Psi}_{\pi\text{-inf}}^{\text{in}}(\xi)$. As a result, optimally curating data under our approximate linear model on policy performance simply entails selecting the least influential demonstrations from the training data \mathcal{D} —in the case of data filtering—or selecting the most influential demonstrations from a new set of demonstrations \mathcal{H} —in the case of data selection:

Task 1: Filter- k Demonstrations

Task 2: Select- k Demonstrations

$$S_{\text{out}}^* = \text{top-}k(\{\Psi_{\pi\text{-inf}}^{\text{out}}(\xi^i) : \xi^i \in \mathcal{D}\}), \quad (4) \quad S_{\text{in}}^* = \text{top-}k(\{\Psi_{\pi\text{-inf}}^{\text{in}}(\xi^i) : \xi^i \in \mathcal{H}\}). \quad (5)$$

We note that by linearly approximating policy performance changes using $\Psi_{\pi\text{-inf}}$, we construct what is commonly termed a (linear) *datamodel* [44]. As shown in NLP [3], using such first-order approximations for data curation can often greatly improve model performance over manual notions of quality.

²Note that the fraction $1/H$ appears from the assumption that all trajectories have equal length, which we make purely for notational simplicity without loss of generality. We refer to the Appendix for the variable length case.

5.3 Additional Quality Metrics

In §5.1, we constructed a method to estimate $\Psi_{\pi\text{-inf}}$ from a dataset of policy rollouts \mathcal{D}_τ by relying on policy gradient methods. Therefore, the estimated performance influence $\widehat{\Psi}_{\pi\text{-inf}}$ becomes increasingly noisy as we reduce the number of rollouts \mathcal{D}_τ to evaluate the policy—akin to the high variance problem of the REINFORCE algorithm. Therefore, to complement the analysis in §5.1, we explore the integration of a *reward agnostic, heuristic* demonstration quality metric based on the action influence scores $\Psi_{a\text{-inf}}$:

$$\Psi_{\text{qual}}(\xi; \mathcal{D}_\tau) := \frac{1}{m} \sum_{\tau \in \mathcal{D}_\tau} \max_{(s,a) \in \tau} \min_{(s',a') \in \xi} \Psi_{a\text{-inf}}((s,a), (s',a')) - \min_{(s,a) \in \tau} \max_{(s',a') \in \xi} \Psi_{a\text{-inf}}((s,a), (s',a')). \quad (6)$$

We base the quality score Eq. 6 on the intuition that we should penalize demonstrations containing outlier or noisy influence scores [13, Sec. 5.2], [10]. Therefore, we posit that this heuristic can reduce variance on tasks requiring precise motion, yet introduce bias uncorrelated with performance in other settings. Thus, in §6, we investigate when the quality score can complement $\Psi_{\pi\text{-inf}}$ to curate data by taking their convex combination, $\alpha \Psi_{\pi\text{-inf}} + (1 - \alpha) \Psi_{\text{qual}}$, ablating $\alpha = 1$ (CUPID) and $\alpha = 1/2$ (CUPID-QUALITY).

6 Experiments

We conduct a series of experiments to test the efficacy of CUPID alongside state-of-the-art baselines for robot data curation. These experiments take place across three simulated tasks from the RoboMimic benchmark suite [45] and three real-world tasks with a Franka FR3 manipulator (see Fig. 4), comprising a taxonomy of settings where data curation may benefit policy performance. We refer to the Appendix for a **detailed description** of our tasks, hardware setup, baselines, and evaluation protocol.

Evaluation. We study the filter- k (Task 1) and select- k (Task 2) curation tasks wherever applicable. For statistical significance, we start filter- k and select- k from random $\sim 2/3$ and $\sim 1/3$ subsets in RoboMimic (300 demonstrations total), and random $\sim 9/10$ and $\sim 4/10$ subsets on Franka tasks (120-160 demonstrations total), respectively. We use the official convolutional-based diffusion policy implementation [46] for all tasks to measure the effect of curation on a state-of-the-art policy architecture. For details on influence function computation for diffusion models, please see the Appendix. We also consider the official π_0 implementation for real-world tasks [19]. To reflect practical constraints, we limit the rollout budget (i.e., the number of rollouts in $\mathcal{D}_\tau = \{\tau^i\}_{i=1}^m$ a curation algorithm may use, as described in §4) to $m = 100$ and $m = 25$ for simulated and real-world tasks, respectively. We report policy success rates over 500 rollouts averaged over the last 10 policy checkpoints for simulated tasks, and 10 rollouts performed with the last checkpoint for real-world tasks.

Baselines. We consider baselines from several methodological categories: DemInf [10]—applicable only to filter- k (Task 1)—curates data offline (i.e., without rollouts) to maximize mutual information, promoting diverse and predictable demonstrations; Demo-SCORE [11] trains binary classifiers to distinguish states from successful and failed rollouts, retaining demonstrations with a high average state success probability; Success Similarity is a custom curation method that measures a demonstration’s average state similarity to successful rollouts, serving as a state-based proxy for CUPID; Random chooses samples uniformly at random; Oracle curates data using ground-truth demonstration labels.

6.1 Setting 1: Improving Policy Performance in Mixed-Quality Regimes

We first study curation of mixed-quality datasets, where training on lower-quality demonstrations may degrade policy performance [45, 10]. We use the “Lift,” “Square,” and “Transport” tasks from RoboMimic’s multi-human (MH) task suite, which provides ground-truth quality labels for demonstrations. On hardware, we design the “Figure-8” task (Fig. 4(a)), where the robot must tie a simplified cleat hitch—a knot that follows a figure-8 pattern—requiring precise manipulation of a deformable rope.

RoboMimic analysis. Fig. 2 presents the RoboMimic benchmark results: the top row shows data quality trends for filter- k and select- k across varying k , while the bottom row reports success rates of diffusion policies trained on the corresponding curated datasets. As expected, we first observe that DemInf—which targets demonstration quality—curates datasets of the highest overall quality by RoboMimic’s ground-truth labels for filter- k (top row, Fig. 2). However, policies trained on data curated by CUPID

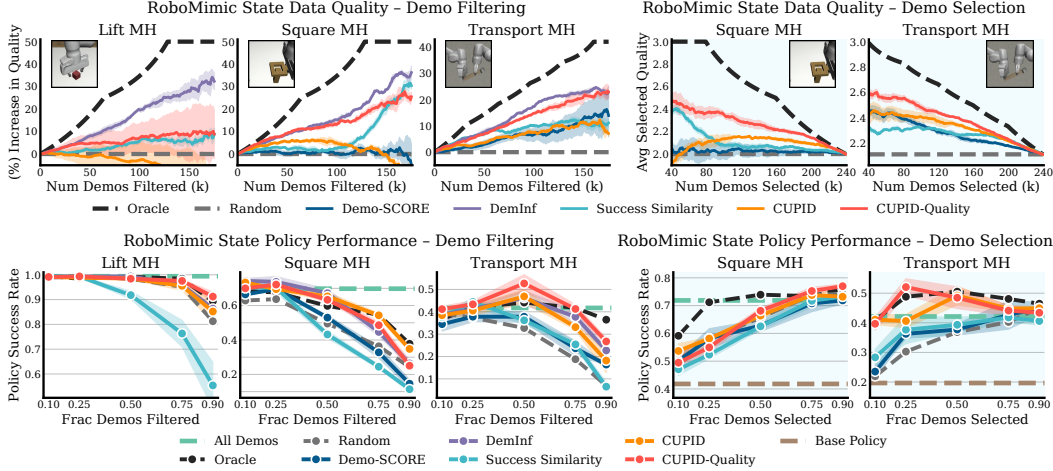


Figure 2: RoboMimic mixed-quality curation results. **Top: Data Quality.** Baselines often prioritize demonstration quality (e.g., DemInf [10]), but highest demonstration quality does not always translate to highest policy success rates. In contrast, CUPID targets demonstrations that most strongly contribute to downstream policy performance. **Bottom: Policy Performance.** Diffusion policies trained on data curated by CUPID achieve higher success rates than baselines, despite using demonstrations of perceived lower quality. Although combining performance and quality measures (CUPID-QUALITY) yields the best policies on mixed-quality datasets, quality measures can degrade performance in other settings (see Fig. 4). Results are averaged over 3 random seeds (500 policies trained across settings). Success rates are computed over 50 rollouts from the last 10 checkpoints (500 rollouts total).

consistently match or outperform those of DemInf (bottom row, Fig. 2). This indicates that human perception of data quality does not necessarily correspond to the data that maximizes downstream policy success. Second, we find that state-based proxies for influence employed by Demo-SCORE [11] and Success Similarity are insufficient in challenging mixed-quality regimes, where successful and failed rollouts contain similar states. Lastly, CUPID-QUALITY, which evenly balances demonstration quality and downstream performance impact (§5.3), attains the highest policy success rates—surpassing the Oracle in 3/5 cases, and achieving an even higher success rate than the official diffusion policy [46] on “Transport MH” while using fewer than (a) 33% of the original 300 demonstrations and (b) 10% of the model parameters. We provide an extended discussion of the RoboMimic results in the Appendix.

Figure-8 analysis. Fig. 4(a) shows diffusion policy results on the real-world “Figure-8” task. First, CUPID improves over the base policy’s success rate by 45% (averaged over filtering and selection). Second, as in RoboMimic, CUPID-QUALITY further strengthens curation performance, corroborating the utility of quality metrics (Eq. 6) in mixed-quality regimes. Finally, Fig. 3(a) demonstrates that the “Figure-8” dataset curated for a single-task diffusion policy (using CUPID-QUALITY) yields an appreciable 45% improvement on the fine-tuned performance of a large, multi-task policy π_0 [19].

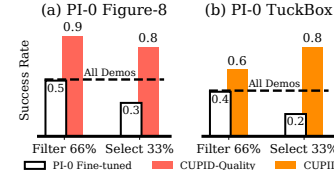


Figure 3: Data curated by single-task diffusion policies improves π_0 [19] post-training performance.

6.2 Setting 2: Identifying Robust Test-time Strategies from Policy Failures

Heterogeneous imitation learning datasets may contain multiple strategies for solving a task, some of which can fail under distribution shifts at deployment. We design a real-world “TuckBox” task, where a robot must tuck a recycling bin under a receptacle by (a) sliding or (b) first repositioning it via pick-and-place (see Fig. 4(b)). The dataset contains a 2:1 ratio of sliding to pick-and-place demonstrations, making sliding the dominant strategy. At test time, we induce an imperceptible distribution shift by altering the bin’s mass distribution, rendering sliding unreliable. In this setting, curation aims to rebalance the dataset to promote strategies that are more robust to unforeseen shifts at deployment.

TuckBox analysis. Fig. 4(b) shows the diffusion policy results on “TuckBox.” Due to the strategy imbalance, the base policy exclusively exhibits the sliding behavior, resulting in a 100% failure rate under the distribution shift. This immediately invalidates the use of Demo-SCORE, which requires both successful and failed rollouts. In contrast, CUPID does not require observing successes: by

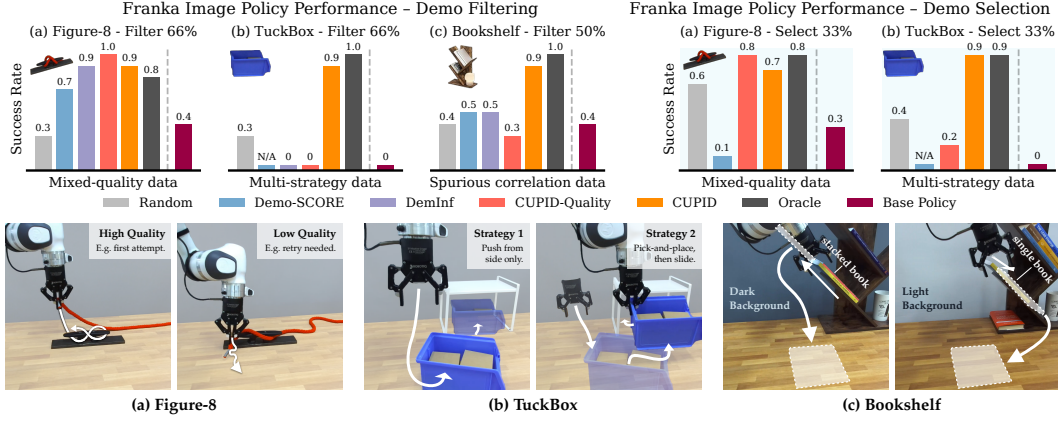


Figure 4: **Franka real-world diffusion policy performance.** CUPID, which curates demonstrations *w.r.t.* policy performance, improves success rates on mixed-quality datasets, identifies robust strategies, and disentangles spurious correlations that hinder performance. Although quality measures (e.g., DemInf, CUPID-QUALITY) help in mixed-quality settings (Figure-8; Fig. 2), they degrade performance when higher-quality demonstrations induce brittle strategies at test time (TuckBox), or when quality is not the primary factor limiting policy success (Bookshelf). Overall, curating data based on performance (CUPID) maintains robustness across these settings.

linking failures to the demonstrations that influenced them, curating with CUPID yields a policy that exhibits increased pick-and-place behavior, performing comparably (90% success rate) to the Oracle. In contrast, both DemInf and CUPID-QUALITY mistakenly conflate the more stochastic pick-and-place demonstrations with low quality, and by removing them, further reinforce the unreliable sliding behavior at deployment. As in §6.1, we conduct an ablation with the π_0 policy (Fig. 3(b)): training on the dataset curated by CUPID for the single-task diffusion policy results in a 40% improvement (averaged over filtering and selection) to π_0 ’s fine-tuned performance on “TuckBox.”

6.3 Setting 3: Disentangling Spurious Correlations in Demonstration Data

Spurious correlations in training data may cause a policy to rely on non-causal features, hindering generalization to variations in the input or task [12]. We design a real-world “Bookshelf” task, where a robot must extract a target book via (a) horizontal or (b) vertical pulling motion, depending on whether another book is stacked above the target. While both strategies are equally represented in the training set, each co-occurs more frequently with a certain background color (see Fig. 4(c)). At evaluation, we test the policy under slight variations in the number and position of distractor books, while keeping the white background fixed—the correlate associated with the horizontal pulling behavior.

Bookshelf analysis. Diffusion policy results are shown in Fig. 4(c). The base policy achieves only a 40% success rate, as the presence of the white background often causes the policy to extract the target book horizontally despite another book being stacked on top (causing it to fall). Interestingly, by training classifiers to distinguish failed from successful states, Demo-SCORE appears to misattribute failure to the presence of rollout correlates (e.g., the stacked book) rather than causal factors (i.e., the white background). In contrast, CUPID attains a 90% success rate by identifying demonstrations that causally drive failure—in this case, horizontal pulling motion with a white background—enabling dataset rebalancing that mitigates the effect of spurious correlations. As in §6.2, DemInf and CUPID-QUALITY incorrectly prioritize higher-quality horizontal pulling motion, leading to negligible performance gains.

7 Conclusion

In this work, we study the problem of data curation for robot imitation learning. We present CUPID, a novel data curation method that uses influence functions to measure the causal impact of a demonstration on the policy’s closed-loop performance. Our results highlight the general utility of performance-based curation for two key curation tasks—filtering existing training demonstrations and subselecting new demonstrations—and across diverse curation settings, where a policy’s test-time performance varies with the choice of training data. We hope this work spurs continued investigation into how training data shapes policy behavior, as is critical to advancing policy reliability and performance in deployment.

8 Limitations

Curation tasks: The curation tasks considered in this work (Task 1 and Task 2) aim to curate performance-maximizing datasets for a specified filtering or selection quantity of demonstrations k . Identifying a suitable quantity of demonstrations to curate represents a possible point of extension.

Data properties: Critically, future work should further investigate how properties of the data dictate the extent to which curation can improve policy performance. For example, our RoboMimic simulation results (Fig. 2) show that while curation benefits performance on “Transport MH” (i.e., a fraction of the demonstrations harm policy performance), removing almost *any* demonstration degrades performance on “Square MH” (i.e., all demonstrations appear important). In contrast, only about 15% of the dataset is necessary to maximize performance on “Lift MH” (i.e., the dataset is highly redundant).

Data explainability: Therefore, while our methods focus on curating existing demonstrations as a first step, future work may seek to interpret the properties influential demonstrations and actively inform subsequent data collection efforts—for example, by providing instructions to collectors.

Selection methods: While the *greedy* selection procedures used in Eq. 4 and Eq. 5 are tractable to optimize and often improve over quality- and similarity-based measures [3], they ignore the interactions between demonstrations in the curated set [14, 44]. This can temper performance gains when the size of the curated set is large. Future work should investigate higher-order approximations that consider the joint diversity of the curated dataset, as is common in the active learning literature (e.g., [47, Sec. 4.3]).

Estimator variance: Finally, although we observe stable performance from CUPID across curation settings, the use of the REINFORCE estimator may result in high variance influence scores, e.g., when the number of policy rollouts is small. In such settings, variance reduction techniques, such as those typically used in reinforcement learning [48], may further improve the fidelity of our influence scores.

References

- [1] R. Grosse, J. Bae, C. Anil, N. Elhage, A. Tamkin, A. Tajdini, B. Steiner, D. Li, E. Durmus, E. Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.
- [2] S. M. Park, K. Georgiev, A. Ilyas, G. Leclerc, and A. Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, pages 27074–27113. PMLR, 2023.
- [3] L. Engstrom, A. Feldmann, and A. Madry. Dsdm: Model-aware dataset selection with datamodels. In *International Conference on Machine Learning*, pages 12491–12526. PMLR, 2024.
- [4] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.
- [5] K. Tirumala, D. Simig, A. Aghajanyan, and A. Morcos. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36:53983–53995, 2023.
- [6] A. Albalak, Y. Elazar, S. M. Xie, S. Longpre, N. Lambert, X. Wang, N. Muennighoff, B. Hou, L. Pan, H. Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.
- [7] A. O’Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.
- [8] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
- [9] S. Kuhar, S. Cheng, S. Chopra, M. Bronars, and D. Xu. Learning to discern: Imitating heterogeneous human demonstrations with preference and representation learning. In *Conference on Robot Learning*, pages 1437–1449. PMLR, 2023.
- [10] J. Hejna, S. Mirchandani, A. Balakrishna, A. Xie, A. Wahid, J. Thompson, P. Sanketi, D. Shah, C. Devin, and D. Sadigh. Robot data curation with mutual information estimators. 2025. URL <https://arxiv.org/abs/2502.08623>.
- [11] A. S. Chen, A. M. Lessing, Y. Liu, and C. Finn. Curating demonstrations using online experience. *arXiv preprint arXiv:2503.03707*, 2025.
- [12] P. De Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.
- [13] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.
- [14] P. W. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019.
- [15] Z. Hammoudeh and D. Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.
- [16] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

- [17] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.025.
- [18] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL <https://proceedings.mlr.press/v229/zitkovich23a.html>.
- [19] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control, 2024. URL <https://arxiv.org/abs/2410.24164>.
- [20] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.
- [21] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. 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. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025.
- [22] J. Gao, A. Xie, T. Xiao, C. Finn, and D. Sadigh. Efficient Data Collection for Robotic Manipulation via Compositional Generalization. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi:10.15607/RSS.2024.XX.013.
- [23] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh. A taxonomy for evaluating generalist robot policies. *arXiv preprint arXiv:2503.01238*, 2025.
- [24] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.
- [25] T. Yu, T. Xiao, A. Stone, J. Thompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv:2302.11550*, 2023.
- [26] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar. Cacti: A framework for scalable multi-task multi-scene visual imitation learning. *arXiv preprint arXiv:2212.05711*, 2022.
- [27] L. Smith, A. Irpan, M. G. Arenas, S. Kirmani, D. Kalashnikov, D. Shah, and T. Xiao. Steer: Flexible robotic manipulation via dense language grounding. *arXiv preprint arXiv:2411.03409*, 2024.

- [28] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 3157–3181. PMLR, 06–09 Nov 2025.
- [29] J. Hejna, C. A. Bhateja, Y. Jiang, K. Pertsch, and D. Sadigh. Remix: Optimizing data mixtures for large scale imitation learning. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 145–164. PMLR, 06–09 Nov 2025.
- [30] H. Shah, S. M. Park, A. Ilyas, and A. Madry. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*, pages 30646–30688. PMLR, 2023.
- [31] A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.
- [32] S. K. Choe, H. Ahn, J. Bae, K. Zhao, M. Kang, Y. Chung, A. Pratapa, W. Neiswanger, E. Strubell, T. Mitamura, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*, 2024.
- [33] M. Xia, S. Malladi, S. Gururangan, S. Arora, and D. Chen. Less: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning*, pages 54104–54132. PMLR, 2024.
- [34] K. Georgiev, R. Rinberg, S. M. Park, S. Garg, A. Ilyas, A. Madry, and S. Neel. Attribute-to-delete: Machine unlearning via datamodel matching. *arXiv preprint arXiv:2410.23232*, 2024.
- [35] S. Basu, P. Pope, and S. Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*, 2021.
- [36] J. Bae, N. Ng, A. Lo, M. Ghassemi, and R. B. Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.
- [37] X. Zheng, T. Pang, C. Du, J. Jiang, and M. Lin. Intriguing properties of data attribution on diffusion models. *arXiv preprint arXiv:2311.00500*, 2023.
- [38] K. Georgiev, J. Vendrow, H. Salman, S. M. Park, and A. Madry. The journey, not the destination: How data guides diffusion models. *arXiv preprint arXiv:2312.06205*, 2023.
- [39] S. Belkhale, Y. Cui, and D. Sadigh. Data quality in imitation learning. *Advances in neural information processing systems*, 36:80375–80395, 2023.
- [40] J. A. Vincent, H. Nishimura, M. Itkina, P. Shah, M. Schwager, and T. Kollar. How generalizable is my behavior cloning policy? a statistical approach to trustworthy performance evaluation. *IEEE Robotics and Automation Letters*, 2024.
- [41] K. Gandhi, S. Karamcheti, M. Liao, and D. Sadigh. Eliciting compatible demonstrations for multi-human imitation learning. In *Conference on Robot Learning*, pages 1981–1991. PMLR, 2023.
- [42] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [43] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

- [44] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Predicting predictions from training data. In *ArXiv preprint arXiv:2202.00622*, 2022.
- [45] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 08–11 Nov 2022.
- [46] 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. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [47] B. Settles. *Active Learning*. Morgan & Claypool Publishers, 2012.
- [48] E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.
- [49] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [50] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- [51] J. Lin, L. Tao, M. Dong, and C. Xu. Diffusion attribution score: Evaluating training data influence in diffusion model. *arXiv preprint arXiv:2410.18639*, 2024.
- [52] B. Mlodozieniec, R. Eschenhagen, J. Bae, A. Immer, D. Krueger, and R. Turner. Influence functions for scalable data attribution in diffusion models. *arXiv preprint arXiv:2410.13850*, 2024.
- [53] T. Xie, H. Li, A. Bai, and C.-J. Hsieh. Data attribution for diffusion models: Timestep-induced bias in influence estimation. *arXiv preprint arXiv:2401.09031*, 2024.
- [54] W. B. Johnson, J. Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.
- [55] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 2003.
- [56] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [57] Y. Dai, J. Lee, N. Fazeli, and J. Chai. Racer: Rich language-guided failure recovery policies for imitation learning. *arXiv preprint arXiv:2409.14674*, 2024.
- [58] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone. Real-time anomaly detection and reactive planning with large language models. In *Robotics: Science and Systems*, 2024.

501 **Appendix Overview – Curating Data your Robot Loves with Influence Functions**

502 The appendix offers additional details with respect to the implementation of CUPID and CUPID-
503 QUALITY (§A), the experiments conducted (§B), along with extended results and analysis (§C), and
504 finally, supporting derivations for our data curation algorithms (§D).

505 **Appendix A Implementation Details** **15**

506 A.1 Influence Functions for Diffusion Policies 15

507 A.2 CUPID Hyperparameters 16

508 A.3 Combining Score Functions 16

509 **Appendix B Experimental Setup** **17**

510 B.1 Hardware Setup 17

511 B.2 Policy Architectures 17

512 B.3 Tasks & Datasets 18

513 B.4 Baseline Details 19

514 **Appendix C Additional Results & Analysis** **20**

515 **Appendix D Derivations** **22**

516 D.1 Proof of Proposition 1 22

517 D.2 Variable Length Derivations 23

A Implementation Details

A.1 Influence Functions for Diffusion Policies

Restatement of Proposition 1. Assume that $\theta(\mathcal{D}) = \operatorname{argmin}_{\theta'} \mathcal{L}_{\text{bc}}(\theta'; \mathcal{D})$, that \mathcal{L}_{bc} is twice differentiable in θ , and that $H_{\text{bc}} \succ 0$ is positive definite (i.e., $\theta(\mathcal{D})$ is not a saddle point)¹. Then, it holds that

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[\frac{R(\tau)}{H} \sum_{(s', a') \in \tau} \sum_{(s, a) \in \xi} \Psi_{a\text{-inf}}((s', a'), (s, a)) \right],$$

where $\Psi_{a\text{-inf}}((s', a'), (s, a)) = -\nabla_{\theta} \log \pi_{\theta}(a'|s') H_{\text{bc}}^{-1} \nabla_{\theta} \ell(s, a; \pi_{\theta})$ is the action influence in Eq. 2.

Computing the Action Influence

Although Proposition 1 provides a clean mechanism to attribute policy performance to its training data by leveraging influence scores on action log-likelihoods, computing $\nabla_{\theta} \log \pi_{\theta}(a'|s')$ (in the action influence $\Psi_{a\text{-inf}}$) for diffusion-based policy architectures is nontrivial due to the iterative denoising process [49, 50]. Instead, various works outside robotics propose to approximate the log-likelihood with the denoising loss $\ell(s', a'; \pi_{\theta})$ for the purpose of data attribution [38], because the denoising loss is proportionate to the variational lower bound on $\log \pi_{\theta}(a'|s')$. In §6, we apply a similar approximation to perform data attribution on state-of-the-art diffusion policies [46], which we describe below.

Diffusion Policy: Consider the standard diffusion policy architecture [46]. An action $a := a^0$ is generated by iteratively denoising an initially random action $a^T \sim \mathcal{N}(0, 1)$ over T steps as a^T, \dots, a^0 using a noise prediction network ϵ_{θ} , where a^i denotes the generated action at the i -th denoising iteration. Following the imitation learning setting described in §4, the parameters θ of the noise prediction network ϵ_{θ} are fit to the BC objective as $\theta = \operatorname{argmin}_{\theta'} \{\mathcal{L}_{\text{bc}}(\theta'; \mathcal{D}) := \frac{1}{|\mathcal{D}|H} \sum_{\xi^i \in \mathcal{D}} \sum_{(s, a) \in \xi^i} \ell(s, a; \pi_{\theta'})\}$. Here, the noise prediction network ϵ_{θ} is trained to predict random noise $\epsilon^i \sim \mathcal{N}(0, 1)$ added to the action a at random timesteps $i \sim \mathcal{U}[0, T)$ of the diffusion process using the loss function ℓ defined as

$$\ell(s, a; \pi_{\theta}) := \mathbb{E}_{\epsilon^i, i} [\|\epsilon^i - \epsilon_{\theta}(\sqrt{\bar{\alpha}_i} a + \sqrt{1 - \bar{\alpha}_i} \epsilon^i, s, i)\|^2], \quad (7)$$

where the constants $\bar{\alpha}_i$ depend on the chosen noise schedule of the diffusion process.

Influence Approximations: Since the denoising loss ℓ in Eq. 7 is proportionate to the variational lower-bound on the action log-likelihood $\log \pi_{\theta}(a|s)$, it may seem intuitive to substitute $\nabla_{\theta} \log \pi_{\theta}(a'|s')$ with $-\nabla_{\theta} \ell(s', a'; \pi_{\theta})$ —assuming gradient alignment—to approximate the action influence (Eq. 2) as

$$\Psi_{a\text{-inf}}((s', a'), (s, a)) \approx \nabla_{\theta} \ell(s', a'; \pi_{\theta}) H_{\text{bc}}^{-1} \nabla_{\theta} \ell(s, a; \pi_{\theta}). \quad (8)$$

A similar approach is taken by Georgiev et al. [38] for attributing the generations of image-based diffusion models. However, consistent with more recent results in the data attribution literature [37, 51], we find this approximation to work poorly in practice, with highly influential training samples $(s, a) \in \mathcal{D}$ rarely reflecting the test-time transitions $(s', a') \in \tau$ over which the action influences are computed. Instead, we follow the approach of Zheng et al. [37], which entails replacing both $\log \pi_{\theta}(a'|s')$ and $\ell(s, a; \pi_{\theta})$ in Eq. 2 with a surrogate, label-agnostic output function $\ell_{\text{square}}(s, a; \pi_{\theta}) := \mathbb{E}_{\epsilon^i, i} [\|\epsilon_{\theta}(\sqrt{\bar{\alpha}_i} a + \sqrt{1 - \bar{\alpha}_i} \epsilon^i, s, i)\|^2]$, making our final approximation of the action influence

$$\Psi_{a\text{-inf}}((s', a'), (s, a)) \approx \nabla_{\theta} \ell_{\text{square}}(s', a'; \pi_{\theta}) H_{\text{square}}^{-1} \nabla_{\theta} \ell_{\text{square}}(s, a; \pi_{\theta}). \quad (9)$$

Here, $H_{\text{square}} = \frac{1}{|\mathcal{D}|H} \sum_{\xi^i \in \mathcal{D}} \sum_{(s, a) \in \xi^i} \nabla_{\theta} \ell_{\text{square}}(s, a; \pi_{\theta}) \nabla_{\theta} \ell_{\text{square}}(s, a; \pi_{\theta})^{\top}$ is the Gauss-Newton approximation of the Hessian—as proposed by Park et al. [2] for more stable and efficient influence estimation—under the surrogate output function ℓ_{square} .

Additional Remarks: While the use of ℓ_{square} may seem counterintuitive at first, it offers three key advantages for computing action influences:

1. Leave-one-out influences (§3) computed using ℓ_{square} (Eq. 9) are (empirically) found to correlate better with actual changes in a diffusion model’s loss—i.e., the difference $\ell(s', a'; \pi_{\theta(\mathcal{D} \setminus (s, a))}) - \ell(s', a'; \pi_{\theta(\mathcal{D})})$ —than those computed using the loss ℓ (Eq. 8) [37].
2. Theoretical analysis also shows that ℓ_{square} more closely aligns with a distributional formulation of the leave-one-out influence compared to the loss ℓ [51]. In the case of diffusion policies, this distributional formulation would seek to design $\Psi_{a\text{-inf}}$ such that it approximates the *leave-one-out divergence* $\Psi_{a\text{-inf}}((s', a'), (s, a)) \approx D_{KL}(\pi_{\theta(\mathcal{D})}(a' | s') || \pi_{\theta(\mathcal{D} \setminus (s, a))}(a' | s'))$.
3. Using ℓ_{square} significantly reduces the computational cost of computing action influences for policies with high-dimensional action spaces, because the ℓ^2 -norm collapses the model’s prediction into a scalar $\|\epsilon_{\theta}(\sqrt{\alpha_i}a + \sqrt{1 - \alpha_i}\epsilon^i, s, i)\|^2$. As a result, computing Eq. 9 requires only a single model gradient $\nabla_{\theta} \ell_{\text{square}}$ per training and test sample. In contrast, while the technique proposed by Lin et al. [51] offers a more accurate estimate of the leave-one-out divergence $\Psi_{a\text{-inf}}((s', a'), (s, a)) \approx D_{KL}(\pi_{\theta(\mathcal{D})}(a' | s') || \pi_{\theta(\mathcal{D} \setminus (s, a))}(a' | s'))$, its computational cost scales linearly with the dimensionality of the model’s output, which may be prohibitive.

Accuracy-Efficiency Tradeoff: We emphasize that our approach for computing the performance influence of a demonstration (Eq. 3) is agnostic to the choice of influence estimation technique [38, 37, 51, 52, 53], allowing practitioners to trade off between accuracy and efficiency based on available computational resources, and enabling integration of improved data attribution methods in the future.

A.2 CUPID Hyperparameters

We use the same set of hyperparameters for CUPID and CUPID-QUALITY across all experiments.

Performance Influence (Eq. 3): For all tasks, we define the trajectory return to be $R(\tau) = 1$ if τ completes the task and $R(\tau) = -1$ otherwise. As a result, every rollout trajectory $\tau \sim p(\cdot | \pi_{\theta})$ provides information on the utility of each demonstration toward the policy’s closed-loop performance. We also found CUPID to work with alternative return definitions—for example, focusing solely on successful rollouts by setting $R(\tau) = 0$ when τ fails. However, such choices may increase sample complexity.

Action Influence (Eq. 9): The action influence requires computing the gradient of an expectation $\nabla_{\theta} \ell_{\text{square}}(s, a; \pi_{\theta}) = \nabla_{\theta} \mathbb{E}_{\epsilon^i, i} [\|\epsilon_{\theta}(\sqrt{\alpha_i}a + \sqrt{1 - \alpha_i}\epsilon^i, s, i)\|^2]$. For all tasks, we approximate the expectation using a batch of $B = 64$ samples $(\epsilon^{(b)}, i^{(b)})$, where $\epsilon^{(b)} \sim \mathcal{N}(0, 1)$ and $i^{(b)} \sim \mathcal{U}[0, T)$ are sampled independently.

Data Attribution: We leverage TRAK [2] to efficiently compute action influences as defined in Eq. 9. First, TRAK uses random projections $\mathbf{P} \sim \mathcal{N}(0, 1)^{p \times d}$, where p is the number of model parameters and $d \ll p$ is the specified projection dimension, to reduce the dimensionality of the gradients as $g_{\theta} = \mathbf{P}^{\top} \nabla_{\theta} \ell_{\text{square}}$ while preserving their inner products $g_{\theta} \cdot g_{\theta} \approx \nabla_{\theta} \ell_{\text{square}} \cdot \nabla_{\theta} \ell_{\text{square}}$ [54]. Second, TRAK ensembles influence scores over C independently trained models (i.e., from different seeds) to account for non-determinism in learning. In our experiments, we use the standard projection dimension $d = 4000$ and minimize computational cost by using only a single policy checkpoint $C = 1$, noting that ensembling over $C > 1$ policy checkpoints is likely to improve the accuracy of our influence scores.

A.3 Combining Score Functions

For ease of exposition in §5.3, we express the overall score of a demonstration as the convex combination of its performance influence and its quality score $\alpha \Psi_{\pi\text{-inf}} + (1 - \alpha) \Psi_{\text{qual}}$, where $\alpha = 1$ and $\alpha \in [0, 1]$ instantiates CUPID and CUPID-QUALITY, respectively. Here, we additionally note that taking weighted combinations of score functions requires first normalizing them to equivalent scales. Hence, our implementation uniformly normalizes demonstration scores within the range $[0, 1]$ (i.e., producing an absolute ranking of demonstrations) for each score function $\Psi_{\pi\text{-inf}}$ and Ψ_{qual} before combining them. This simple approach can be applied to combine an arbitrary number of demonstration score functions.

B Experimental Setup

B.1 Hardware Setup

As depicted in Fig. 4, our hardware experiments involve a Franka FR3 manipulator robot. We use a single ZED 2 camera to capture RGB-D observations and disregard the depth information. Our image-based policies process the RGB observations (after downsampling to (256, 256)) and predict sequences of end-effector poses for the manipulator, which are tracked using operational space control [55].

B.2 Policy Architectures

Diffusion Policy: We use the original diffusion policy implementation³ from Chi et al. [46]. Specifically, we use the convolutional-based diffusion policy architecture for efficiency. For state-based tasks (e.g., in RoboMimic; Fig. 2), actions are generated solely using the noise prediction network ϵ_θ as described in §A.1. However, for image-based tasks (e.g., on hardware; Fig. 4), the policy π_θ contains two sets of parameters $\theta = (\theta_o, \theta_a)$ corresponding to a ResNet-18 encoder E_{θ_o} and the noise prediction network ϵ_{θ_a} . When scoring demonstrations, we compute action influences (Eq. 9) over all available policy parameters θ , noting that one might also consider using a subset of the parameters, e.g., those of the noise prediction network or an alternative action head, under reduced computational budgets.

Other optimizations: In preliminary experiments, we found that the original diffusion policy (a) was heavily over-parameterized and (b) converged in performance much earlier in training than the specified maximum number of epochs. Thus, to accelerate experimentation in RoboMimic (Fig. 2), we (a) manually determined the smallest model size that performed similarly to the original policy and (b) adjusted the maximum number of epochs to the point where additional training would result in no further performance gains. Importantly, we keep the model size and training epochs consistent across all curation methods for a given RoboMimic task. For real-world hardware experiments, we use the same model size and limit the number of training steps to 200K across all tasks, similar to Hejna et al. [10]. All other diffusion policy hyperparameters are consistent with the original implementation [46].

Generalist Policy (π_0): We finetune Physical Intelligence’s PI-0 vision-language action (VLA) policy [19] via Low-Rank Adaptation (LoRA) [56] on the “Figure-8” and “TuckBox” tasks. To ensure the post-trained policy’s performance is solely a result of the properties of the curated dataset used for training, we use the standard fine-tuning parameter configuration recommended by Black et al. [19] and keep all hyperparameters fixed for all experiments. We list these parameters in Table 1. We trained on 2 NVIDIA RTX 4090 GPUs, which took approximately 15 hours under the configuration in Table 1. In initial experiments, we found that training for 30K steps was necessary to compensate for mismatch between our robot’s action space (target end-effector poses tracked via operational space control) and the action spaces used to train the base PI-0 policy (absolute joint angles). In addition, we found that using a descriptive prompt for the task was necessary to yield performant policies, even though we fine-tune the base policy for single-task use. We kept these prompts fixed across training, evaluation, and all curation settings. For “TuckBox,” we used the instruction “Move the blue box underneath the white shelf” to avoid biasing the policy towards a particular behavior mode (e.g., “pushing” or “picking”). For “Figure-8,” we used the instruction “Pick up the red rope, then tie a figure 8” where we found the two-step instruction to increase performance over simpler instructions like “Tie the cleat.” Similar to the diffusion policy experiment, we fine-tune a separate PI-0 model for each curation task, filter- k (Task 1) and select- k (Task 2), using their corresponding base demonstration datasets. Then, we fine-tune additional PI-0 models on datasets curated by our methods for diffusion policies (Fig. 4).

³Open-source implementation at https://github.com/real-stanford/diffusion_policy.

Parameter	Value
Train Steps	30000
Batch Size	16
Optimizer	AdamW
LR Schedule	Cosine Decay
EMA	False
Action Chunk Length	50 steps
Control Frequency	10 Hz
Image Resolution	(224, 224)
Observation History	1
VLM Backbone LoRA	Rank=16, $\alpha = 16$
Action Expert LoRA	Rank=32, $\alpha = 32$

Table 1: Hyperparameter configuration for PI-0 fine-tuning.

B.3 Tasks & Datasets

Here, we provide additional details regarding our real-world hardware tasks and their corresponding datasets. We refer to Mandlekar et al. [45] for details on the simulated RoboMimic benchmark.

Figure-8: A brief description of the task is provided in §6.1. The “Figure-8” dataset contains 160 demonstrations evenly split across four *quality tiers*. Higher quality demonstrations complete the task at a constant rate without errors, while lower-quality demonstrations vary in progression rate [?] and include retry or recovery behaviors. Therefore, the “Figure-8” task intends to reflect a practical setting where demonstrations of varying properties are introduced during data collection, whether organically or deliberately, e.g., to improve policy robustness to recoverable failures [57]. Therefore, we expect curation algorithms that distinguish demonstrations upon notions of quality (e.g., predictability [10]) to perform well on this task, which is consistent with our findings in Fig. 4(a) and Fig. 3(a).

TuckBox: A brief description of the task is provided in §6.2. As mentioned, the “TuckBox” dataset contains 120 demonstrations split 2:1 between two subsets: 80 demonstrations solve the task by sliding the box under the receptacle, while 40 demonstrations first reposition the box in front of the receptacle via pick-and-place. Although the sliding strategy appears more smooth and involves just a single step, it is rendered unreliable by imperceptible test-time distribution shifts to the box’s mass distribution. In essence, “TuckBox” stands conceptually opposite to “Figure-8,” whereby attending to heuristic properties of the demonstrations may result in poor curation performance (as shown in Fig. 4(b)).

Bookshelf: A brief description of the task is provided in §6.3. To summarize, the robot must extract a target book that is either shelved alone—affording a simple, horizontal pulling motion—or with another book stacked on top of it (i.e., a *bookstack*). In the bookstack case, the robot must extract the target book using a vertical pulling motion, such that the stacked book does not fall off the shelf in the process (see Fig. 4(c)). In total, the “Bookshelf” dataset contains 120 demonstrations split across three subsets: (a) 60 demonstrations feature the target book shelved alone with a white background, (b) 20 demonstrations feature the bookstack with a white background, and (c) 40 demonstrations feature the bookstack with a dark background. All subsets feature task-irrelevant distractor books on other shelves.

Spurious correlations in training data: Although the vertical pulling solution to the bookstack case is demonstrated in scenes with both white and dark backgrounds, the disproportionate number of demonstrations in subset (a) versus subset (b) spuriously correlates the horizontal pulling motion with the white background. Such spurious correlations may result in *causal confusion* [12], where the policy ignores the bookstack, attends the white background, and executes the failing horizontal strategy.

Spurious correlations in rollout data: Like “TuckBox,” “Bookshelf” represents another limiting case for curating data with quality metrics [10]. However, it also presents an additional challenge for methods that seek to curate data using online experience [11]. Namely, we highlight that attending to differences in states between successful and failed policy rollouts may be susceptible to spurious

677 correlations in the rollout data. Consider the simple case: if we were to observe successful rollouts
 678 when the target book is shelved alone and failed rollouts when another book is stacked above the target,
 679 then training a classifier (i.e., as in Demo-SCORE [11]) to distinguish successful from failed states
 680 may wrongly attribute failures to the presence of the stacked book. Curating demonstrations with such
 681 a classifier would, in turn, worsen the spurious correlation in the training data. Beyond this simple case,
 682 we posit that handling more challenging instances in real-world settings requires methods that *causally*
 683 *attribute* the outcomes of observed test-time experiences to the training data, such as CUPID.

684 B.4 Baseline Details

685 **DemInf:** We use the official implementation⁴ provided by the authors [10]. We note that DemInf
 686 curates data offline—that is, without using any policy rollouts—and is at present only applicable to the
 687 demonstration filtering setting (i.e., filter- k , as defined in Task 1).

688 **Demo-SCORE:** We construct our own implementation based on the description provided by the
 689 authors [11]. Given our assumed fixed budget of $m = 100$ rollouts for RoboMimic experiments (§6),
 690 we collect 25 rollouts from $C = 4$ policy checkpoints throughout training. We train three-layer MLP
 691 classifiers with hidden dimensions [16,16,16] on the first three rollout sets, and select the best classifier
 692 via cross-validation on the last 25 rollouts, as described in [11]. Since we reduce the rollout budget
 693 to $m = 25$ rollouts for hardware experiments (§6), we collect 25 rollouts from the last $C = 1$ policy
 694 checkpoint. We then train a single ResNet-18 encoder and three-layer classification head with hidden
 695 dimensions [32,32,32] on 20 of the rollouts, leaving 5 validation rollouts to monitor for overfitting. We
 696 train all classifiers with a heavy dropout of 0.3 and an AdamW weight decay of 0.1 to prevent overfitting,
 697 in alignment with [11]. Although Chen et al. [11] only test Demo-SCORE for demonstration filtering,
 698 we extend its use for demonstration selection (i.e., select- k , as defined in Task 2).

699 **Success Similarity:** We design a custom robot data curation algorithm that, similar to Demo-SCORE,
 700 evaluates demonstrations based on a heuristic measure of similarity *w.r.t.* successful policy rollouts.
 701 Instead of training classifiers, Success Similarity measures the average state-embedding similarity of a
 702 demonstration *w.r.t.* all successful rollouts as

$$S(\xi; \mathcal{D}_\tau) = - \sum_{\tau \in \mathcal{D}_\tau} \left[\mathbf{1}(R(\tau) = 1) \cdot \frac{1}{H^2} \sum_{s' \in \tau} \sum_{s \in \xi} D(s', s; \pi_\theta) \right],$$

703 where the indicator function $\mathbf{1}$ evaluates to 1 if rollout τ is successful and 0 otherwise, H is the assumed
 704 length of all demonstrations $\xi \in \mathcal{D}$ and rollouts $\tau \in \mathcal{D}_\tau$ for notational simplicity, and D is a specified
 705 distance function over states [58], such as the Mahalanobis, L2, or cosine distance. We found multiple
 706 distance functions to work well, and ultimately used the L2 distance in our experiments.

707 **Comparison to Performance Influence (CUPID):** One can interpret Success Similarity as replacing the
 708 action influence $\Psi_{a-\text{inf}}((s', a'), (s, a))$ (Eq. 2) with a state-based proxy $-D(s', s; \pi_\theta)$ in an attempt to
 709 estimate the performance contribution of a demonstration (Eq. 3). In our RoboMimic experiments
 710 (Fig. 2), this approach performs comparably to Demo-SCORE and, in some cases, even outperforms
 711 it—without requiring any model training. However, Success Similarity performs consistently worse
 712 than CUPID across all tasks, supporting prior findings that influence functions offer a substantially
 713 stronger causal signal than heuristic measures of similarity [2].

714 **Oracle:** For each task, the Oracle method represents a best attempt to curate data assuming privileged
 715 access to ground-truth demonstration labels. For the RoboMimic and “Figure-8” tasks, the Oracle
 716 ranks demonstrations in descending order of quality, choosing high-quality demonstrations before
 717 low-quality demonstrations. For the “TuckBox” task, the Oracle first chooses all demonstrations
 718 exhibiting the more robust pick-and-place strategy before any demonstration exhibiting the more brittle
 719 sliding strategy. Lastly, for the “Bookshelf” task, the Oracle chooses demonstrations to minimize the
 720 effect of the *known* spurious correlation (i.e., horizontal pulling motion in the presence of a white

⁴Open-source implementation at <https://github.com/jhejna/demonstration-information>.

background), resulting in a more balanced curated dataset. These definitions of the Oracle apply to both the filter- k (Task 1) and select- k (Task 2) curation tasks studied throughout this work.

Additional baselines: We implement a number of additional custom baselines that one might try in practice, such as curating data based on policy loss, policy uncertainty, state diversity, and action diversity. However, we exclude them from our experiments given their relatively poor performance. We will provide code that contains the implementation of all methods upon potential acceptance.

C Additional Results & Analysis

We provide an extended discussion on our RoboMimic results (Fig. 2), along with additional results and ablations for RoboMimic and π_0 (Fig. 3) that were cut from the main text due to space constraints.

Discussion: How is curation performance affected by properties of the data and the task?

Performance versus Data Quality: A key finding we emphasize is that the performance of a state-of-the-art policy does not necessarily correlate with the *perceived quality* of its training data. Factors such as redundancy, balance, and coverage of the dataset all play a role in determining the final performance of a policy. This is illustrated in the Oracle filter- k results (left three plots of Fig. 2). While the top row shows a monotonic increase in average dataset quality as lower-quality demonstrations are filtered out, the bottom row reveals (1) a consistent performance drop for diffusion policies on 2 out of 3 tasks, and (2) as expected, performance degradation when too many demonstrations are removed. Similar analysis applies to the select- k setting. These results highlight two important points: First, the impact of dataset curation should not be judged by quality labels alone, but by the downstream performance of models trained on curated datasets. Second, determining how much data to curate (i.e., the k in filter- k and select- k) remains another key challenge for effective data curation in practice.

Performance versus Task Complexity: We further study how curation performance varies with task complexity by evaluating three RoboMimic tasks of increasing difficulty—“Lift MH,” “Square MH,” and “Transport MH.” As shown in the bottom row of Fig. 2, diffusion policies achieve 100% success on the easiest task, “Lift MH,” even when trained on all demonstrations, indicating that low-quality demonstrations have little to no impact⁵. Consequently, many demonstrations can be filtered without affecting policy performance. We see a similar trend for the moderately difficult “Square MH” task, where the policy benefits from access to all demonstrations regardless of their quality. However, performance degrades more quickly as demonstrations are filtered, suggesting increased sensitivity to data quantity due to the task’s higher complexity relative to “Lift MH.” Finally, for the most challenging task, “Transport MH,” which requires precise bi-manual coordination, both CUPID and CUPID-QUALITY yield clear performance gains over the base policy. In sum, these results suggest that curation of mixed-quality datasets is most beneficial for complex, precision-critical tasks, where low-quality demonstrations are more likely to degrade policy performance.

Ablation: How do CUPID’s influence estimates vary with the number of policy rollouts?

We conduct an ablation study in RoboMimic evaluating the quality of datasets curated by CUPID and CUPID-QUALITY under varying numbers of rollouts, $m \in \{1, 5, 10, 25, 50, 100\}$. The results for state-based and image-based diffusion policies are shown in Fig. 5 and Fig. 6, respectively. For the “Lift MH” and “Square MH” tasks, performance influences (Eq. 3) stabilize around $m \in [25, 50]$, yielding quality trends similar to those obtained with $m = 100$. In contrast, for the more challenging “Transport MH” task, quality trends continue to evolve with increasing rollouts, suggesting that more rollouts are required to obtain reliable influence estimates in complex task settings, where curation matters most.

⁵Note that Fig. 2 does not include select- k curation results for “Lift MH” because the base policy already achieves a 100% success rate, leaving no further room for improvement by selecting additional demonstrations.

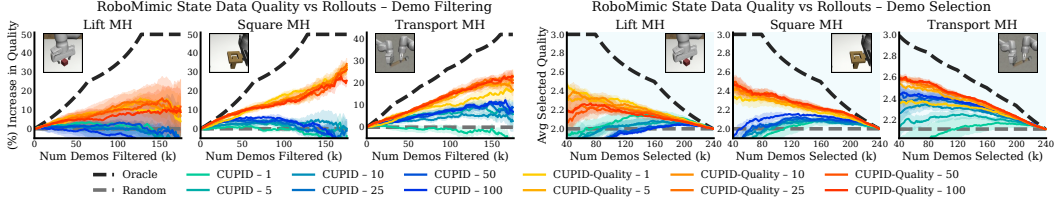


Figure 5: RoboMimic state ablation: Data quality trends under varying number of rollouts. Performance influences (Eq. 3) appear to converge around $m \in [25, 50]$ rollouts for “Lift MH” and “Square MH” (yielding similar quality trends), but continue to evolve with more rollouts for “Transport MH.” Curation performed on state-based diffusion policies. Results are averaged over 3 random seeds. Errors bars represent the standard error.

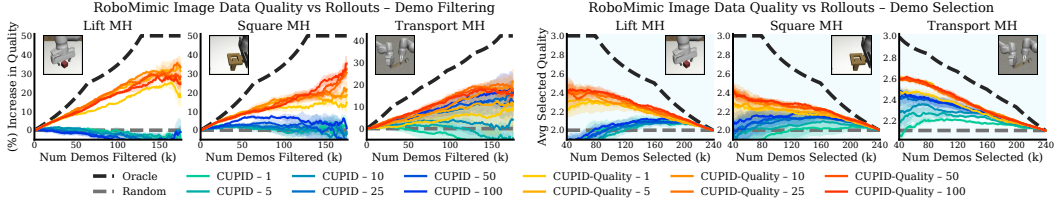


Figure 6: RoboMimic image ablation: Data quality trends under varying number of rollouts. Performance influences (Eq. 3) appear to converge around $m \in [25, 50]$ rollouts for “Lift MH” and “Square MH” (yielding similar quality trends), but continue to evolve with more rollouts for “Transport MH.” Curation performed on image-based diffusion policies. Results are averaged over 3 random seeds. Errors bars represent the standard error.

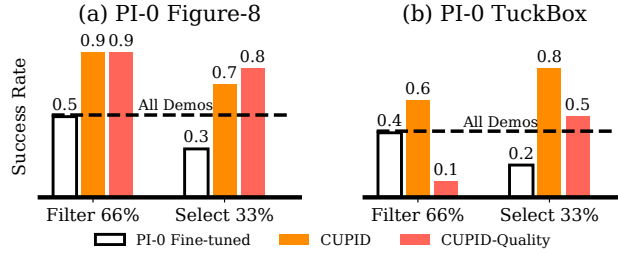


Figure 7: Data curated by single-task diffusion policies improves π_0 [19] post-training performance. As in Fig. 4, quality measures (CUPID-QUALITY) may degrade performance when higher-quality demonstrations induce brittle strategies at test time (TuckBox), whereas curating based on performance (CUPID) remains robust across settings.

763 Additional results & analysis: π_0 policy performance

764 Fig. 7 contains the full results of our π_0 ablation (Fig. 3), including the performance of π_0 [19] trained
 765 on datasets curated by CUPID and CUPID-QUALITY for both the “Figure-8” and “TuckBox” tasks.

766 In this experiment, we investigate two questions: (1) Can datasets curated with one policy architecture
 767 result in increased performance when used to train another policy with a different architecture? (2)
 768 How influential is curation for policies that have been pre-trained on large-scale multi-task datasets?

769 *Curation Transfer:* Towards the first question, Fig. 7 shows that the datasets curated using the diffusion
 770 policy significantly increase performance of the fine-tuned PI-0 policies relative to fine-tuning on the
 771 base datasets. We attribute these results to two causes: First, we find that both the diffusion policy
 772 and PI-0 have sufficient capacity to accurately fit the training data distribution, and thus, they must
 773 learn a similar behavior distribution from the training data. This implies that the observed performance
 774 gains in Fig. 7 result from curation transfer between policies. Second, as the “TuckBox” experiment
 775 shows (Fig. 4(b)), our method is able to effectively identify behaviors in the demonstration data that are
 776 not robust. While on-policy evaluations (i.e., rollouts) are necessary to identify such brittle behaviors,
 777 these are purely properties of the training demonstration data. Therefore, filtering out poor behaviors
 778 will increase performance for any policy. Similarly, on the high-precision “Figure-8” task, filtering out
 779 more noisy, low-quality demonstrations is likely to improve performance for any policy.

780 *VLA Robustness*: Towards the second question, we find that even when the base policy is pre-trained on
 781 a large, diverse, multi-task dataset, curation is still essential to yield strong fine-tuned performance. As
 782 shown in Fig. 7, PI-0 policies trained on the base demonstration datasets are unable to reliably complete
 783 our tasks. In contrast, policies trained on curated datasets attain significantly higher success rates. As
 784 such, our results indicate that simply training VLM-based policies on more data and more tasks does
 785 not strictly result in pre-conditioned policies that use their generalist knowledge to “ignore” low-quality
 786 behaviors or brittle strategies in demonstration data—i.e., data curation still appears essential.

787 *Concluding Remarks*: Overall, these results indicate that using smaller, single-task policies to curate
 788 individual datasets, which may then benefit a larger, multi-task policy is a promising direction to
 789 alleviate the computational cost of applying our method to generalist policies. Still, we emphasize
 790 that datasets curated using our method are not completely *model agnostic*, implying that samples in
 791 the demonstration data may affect the models differently. As such, while PI-0 achieves a higher base
 792 performance than the diffusion policy, the PI-0 policies trained on curated datasets perform similarly
 793 to or slightly worse than the diffusion policies (for which those datasets were curated).

794 Additional results: RoboMimic data quality

795 We provide full data quality results in RoboMimic. Fig. 8 is identical to the top row of Fig. 2 in the main
 796 text, but also includes data quality trends for select- k curation on “Lift MH.” Fig. 9 shows data quality
 797 results for image-based diffusion policies. Note that we do not retrain image-based policies on curated
 798 datasets (as in the bottom row of Fig. 2) due to the substantial computational resources required.

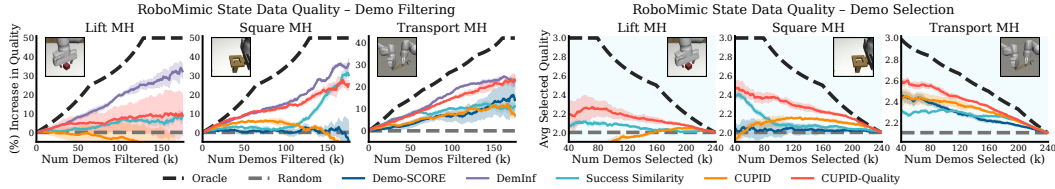


Figure 8: RoboMimic state data quality results. Curation performed on state-based diffusion policies. Results are averaged over 3 random seeds. Errors bars represent the standard error.

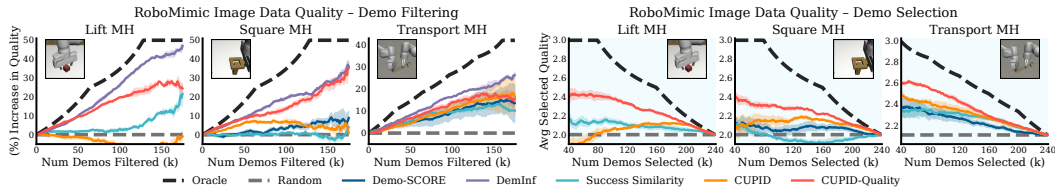


Figure 9: RoboMimic image data quality results. Curation performed on image-based diffusion policies. Results are averaged over 3 random seeds. Errors bars represent the standard error.

799 D Derivations

800 D.1 Proof of Proposition 1

801 *Proof.* As presented in §3, applying the basic derivation of the influence function¹ in [13] gives us that

$$\begin{aligned}\Psi_{\pi\text{-inf}}(\xi) &:= \left. \frac{dJ(\pi_\theta)}{d\epsilon} \right|_{\epsilon=0} \\ &= -\nabla_\theta J(\pi_\theta) \nabla_\theta^2 \mathcal{L}_{bc}(\theta; \mathcal{D})^{-1} \nabla_\theta \mathcal{L}_{traj}(\xi; \pi_\theta).\end{aligned}$$

802 Next, note that the standard log-derivative trick underlying policy gradient methods [16, 43] tells us
 803 that

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[R(\tau) \sum_{(s', a') \in \tau} \nabla_\theta \log \pi(a'|s') \right].$$

Therefore, since \mathcal{L}_{bc} and $\mathcal{L}_{\text{traj}}$ are deterministic functions of θ , ξ , and \mathcal{D} , it holds that

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[R(\tau) \sum_{(s',a') \in \tau} -\nabla_\theta \log \pi_\theta(a'|s') H_{\text{bc}}^{-1} \nabla_\theta \mathcal{L}_{\text{traj}}(\xi; \pi_\theta) \right]$$

by linearity of expectation. Finally, by simply noting that $\mathcal{L}_{\text{traj}}(\xi; \pi_\theta) = \frac{1}{H} \sum_{(s,a) \in \xi} \ell(s,a;\theta)$ and applying the definition of $\Psi_{a\text{-inf}}$, we have the result:

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[\frac{R(\tau)}{H} \sum_{(s',a') \in \tau} \sum_{(s,a) \in \xi} \Psi_{a\text{-inf}}((s',a'),(s,a)) \right].$$

□

D.2 Variable Length Derivations

In §4 and §5, we assumed that all trajectories in the demonstration dataset \mathcal{D} were of an equal length H for notational simplicity. Here, we show that without loss of generality, our analysis extends to the case where the length of demonstration trajectories vary. Suppose each demonstration $\xi^i \in \mathcal{D}$ has length H^i , so that the base policy π_θ minimizes the average loss across all samples in the demonstration data, i.e.,

$$\theta = \operatorname{argmin}_{\theta'} \{ \tilde{\mathcal{L}}_{\text{bc}}(\theta'; \mathcal{D}) := \frac{1}{(\sum_{i=1}^n H^i)} \sum_{\xi^i \in \mathcal{D}} \sum_{(s,a) \in \xi^i} \ell(s,a;\pi_{\theta'}) \}. \quad (10)$$

Note that the objective in Eq. 10 is equivalent to an unweighted BC loss

$$\mathcal{L}'_{\text{bc}}(\theta'; \mathcal{D}) := \sum_{\xi^i \in \mathcal{D}} \sum_{(s,a) \in \xi^i} \ell(s,a;\pi_{\theta'}),$$

which decomposes into its unweighted trajectory losses $\mathcal{L}'_{\text{traj}}(\xi; \pi_\theta) := \sum_{(s,a) \in \xi} \ell(s,a;\pi_\theta)$, so that $\mathcal{L}'_{\text{bc}}(\theta; \mathcal{D}) = \sum_{\xi^i \in \mathcal{D}} \mathcal{L}'_{\text{traj}}(\xi^i; \pi_\theta)$. We can then derive an equivalent statement to Proposition 1 for the unweighted loss functions that applies when the demonstrations have variable length.

Proposition 2. Assume that $\theta(\mathcal{D}) = \operatorname{argmin}_{\theta'} \mathcal{L}'_{\text{bc}}(\theta'; \mathcal{D})$, that \mathcal{L}'_{bc} is twice differentiable in θ , and that $H_{\text{bc}} \succ 0$ is positive definite (i.e., $\theta(\mathcal{D})$ is not a saddle point)¹. Then, it holds that

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[R(\tau) \sum_{(s',a') \in \tau} \sum_{(s,a) \in \xi} \Psi_{a\text{-inf}}((s',a'),(s,a)) \right]. \quad (11)$$

Proof. As presented in §3, applying the basic derivation of the influence function¹ in [13] gives us that

$$\begin{aligned} \Psi_{\pi\text{-inf}}(\xi) &:= \left. \frac{dJ(\pi_\theta)}{d\epsilon} \right|_{\epsilon=0} \\ &= -\nabla_\theta J(\pi_\theta) \nabla_\theta^2 \mathcal{L}'_{\text{bc}}(\theta; \mathcal{D})^{-1} \nabla_\theta \mathcal{L}'_{\text{traj}}(\xi; \pi_\theta). \end{aligned}$$

Next, note that the standard log-derivative trick underlying policy gradient methods [16, 43] tells us that

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[R(\tau) \sum_{(s',a') \in \tau} \nabla_\theta \log \pi(a'|s') \right].$$

Therefore, since \mathcal{L}'_{bc} and $\mathcal{L}'_{\text{traj}}$ are deterministic functions of θ , ξ , and \mathcal{D} , it holds that

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[R(\tau) \sum_{(s',a') \in \tau} -\nabla_\theta \log \pi_\theta(a'|s') H_{\text{bc}}^{-1} \nabla_\theta \mathcal{L}'_{\text{traj}}(\xi; \pi_\theta) \right]$$

by linearity of expectation. Finally, by simply noting that $\mathcal{L}'_{\text{traj}}(\xi; \pi_\theta) = \sum_{(s,a) \in \xi} \ell(s,a;\theta)$ and applying the definition of $\Psi_{a\text{-inf}}$, we have the result:

$$\Psi_{\pi\text{-inf}}(\xi) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta)} \left[R(\tau) \sum_{(s',a') \in \tau} \sum_{(s,a) \in \xi} \Psi_{a\text{-inf}}((s',a'),(s,a)) \right].$$

□