

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

518 A Implementation Details

519 A.1 Influence Functions for Diffusion Policies

520 **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 differ-
 521 entiable 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
 522 that

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

523 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.

524 Computing the Action Influence

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

532 **Diffusion Policy:** Consider the standard diffusion policy architecture [46]. An action $a := a^0$ is
 533 generated by iteratively denoising an initially random action $a^T \sim \mathcal{N}(0, 1)$ over T steps as a^T, \dots, a^0
 534 using a noise prediction network ϵ_{θ} , where a^i denotes the generated action at the i -th denoising iteration.
 535 Following the imitation learning setting described in §4, the parameters θ of the noise prediction network
 536 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,
 537 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
 538 random timesteps $i \sim \mathcal{U}[0, T)$ of the diffusion process using the loss function ℓ defined as

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

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

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

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

543 A similar approach is taken by Georgiev et al. [38] for attributing the generations of image-based
 544 diffusion models. However, consistent with more recent results in the data attribution literature [37, 51],
 545 we find this approximation to work poorly in practice, with highly influential training samples $(s, a) \in \mathcal{D}$
 546 rarely reflecting the test-time transitions $(s', a') \in \tau$ over which the action influences are computed.
 547 Instead, we follow the approach of Zheng et al. [37], which entails replacing both $\log \pi_{\theta}(a'|s')$ and
 548 $\ell(s, a; \pi_{\theta})$ in Eq. 2 with a surrogate, label-agnostic output function $\ell_{\text{square}}(s, a; \pi_{\theta}) := \mathbb{E}_{\epsilon^i, i} [\|\epsilon_{\theta}(\sqrt{\bar{\alpha}_i} a +$
 549 $\sqrt{1 - \bar{\alpha}_i} \epsilon^i, s, i)\|^2]$, making our final approximation of the action influence

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

550 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
 551 approximation of the Hessian—as proposed by Park et al. [2] for more stable and efficient influence
 552 estimation—under the surrogate output function ℓ_{square} .

553 **Additional Remarks:** While the use of ℓ_{square} may seem counterintuitive at first, it offers three key
 554 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.

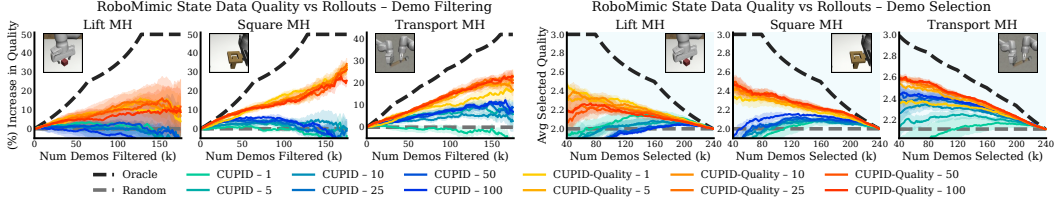


Figure 5: RoboMimic state ablation: Data quality trends under varying number of rollouts. Performance influences (Eq. 3) appear to converge around $m \in [25, 50]$ rollouts for “Lift MH” and “Square MH” (yielding similar quality trends), but continue to evolve with more rollouts for “Transport MH.” Curation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

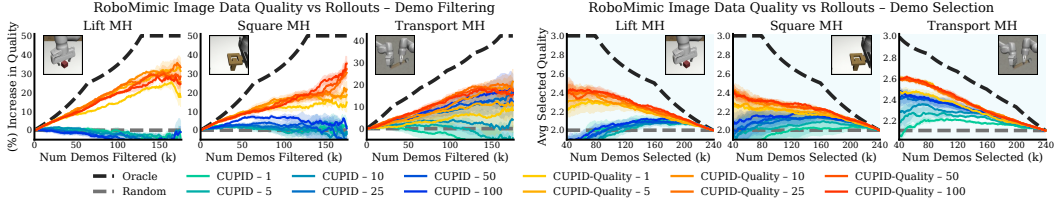


Figure 6: RoboMimic image ablation: Data quality trends under varying number of rollouts. Performance influences (Eq. 3) appear to converge around $m \in [25, 50]$ rollouts for “Lift MH” and “Square MH” (yielding similar quality trends), but continue to evolve with more rollouts for “Transport MH.” Curation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

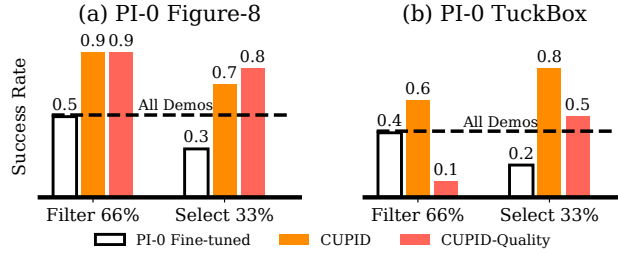


Figure 7: Data curated by single-task diffusion policies improves π_0 [19] post-training performance. As in Fig. 4, quality measures (CUPID-QUALITY) may degrade performance when higher-quality demonstrations induce brittle strategies at test time (TuckBox), whereas curating based on performance (CUPID) 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 and
 772 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.

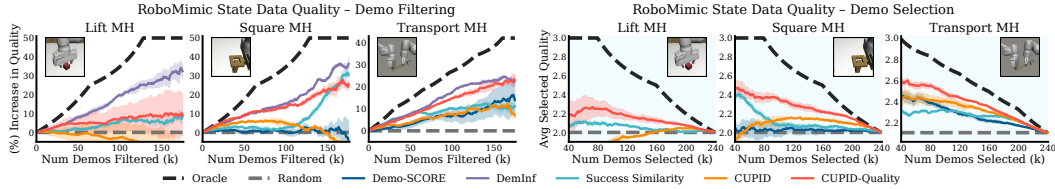


Figure 8: RoboMimic state data quality results. Curation performed on state-based diffusion policies. Results are averaged over 3 random seeds. Errors bars represent the standard error.

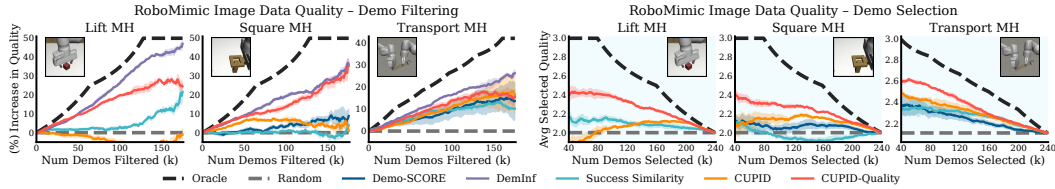


Figure 9: RoboMimic image data quality results. Curation performed on image-based diffusion policies. Results are averaged over 3 random seeds. Errors bars represent the standard error.

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:

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

□