SCIZOR: A Self-Supervised Approach to Data Curation for Large-Scale Imitation Learning

Yu Zhang[1*], **Yuqi Xie**[1,2*], **Huihan Liu**[1†], **Rutav Shah**[1†],
Michael Wan[‡2], **Linxi "Jim" Fan**[2], **Yuke Zhu**[1,2]
[1]The University of Texas at Austin [2]NVIDIA Research

Abstract: Imitation learning advances robot capabilities by enabling the acquisition of diverse behaviors from human demonstrations. However, large-scale datasets used for policy training often introduce substantial variability in quality, which can negatively impact performance. As a result, automatically curating datasets by filtering low-quality samples to improve quality becomes essential. Existing robotic curation approaches rely on costly manual annotations and perform curation at a coarse granularity, such as the dataset or trajectory level, failing to account for the quality of individual state-action pairs. To address this, we introduce SCIZOR, a self-supervised data curation framework that filters out low-quality state-action pairs to improve the performance of imitation learning policies. SCIZOR targets two complementary sources of low-quality data: *suboptimal* data, which hinder learning with undesirable actions, and *redundant* data, which dilute training with repetitive patterns. SCIZOR leverages a self-supervised task progress predictor for suboptimal data to remove samples lacking task progression, and a deduplication module operating on joint state-action representation for samples with redundant patterns. Empirically, we show that SCIZOR enables imitation learning policies to achieve higher performance with less data, yielding an average improvement of 15.4% across multiple benchmarks. More information is available at: https://ut-austin-rpl.github.io/SCIZOR/

Keywords: Imitation Learning, Data Curation, Robot Foundation Models

1 Introduction

Imitation learning has shown promising signs in acquiring a wide range of motor behaviors by learning from expert demonstrations, a necessary step towards general-purpose robots. The success of the imitation learning hinges on the extensive datasets collected in a wide range of tasks and environments, often collected by different demonstrators. Such diverse, large-scale data collection inherently introduces variability in data quality [1, 2, 3], including mistakes made by operators leading to suboptimal actions (e.g., dropping an object), or redundancy in data leading to skewed distributions. Such a dataset can misguide models into learning incorrect behaviors [1, 4] and hinder diversity [3], reducing the impact of rare but informative actions. Therefore, effective data curation — the process of filtering data to improve the data quality [5, 6] — becomes critical for building robust and high-performing imitation learning policies.

Early efforts in robotic data curation have relied heavily on human annotations to label high- and low-quality data [1], but these methods have largely been confined to small-scale datasets. As data scales up, manual annotation becomes infeasible, making it important to *automatically* curate data, an approach that has already shown promise in fields like computer vision (CV) and natural language processing (NLP) [7, 8, 9, 10]. Specifically in robot learning, a key challenge in this process

[*, †] Equal contribution.
[‡] This work was done while Michael Wan was interning at NVIDIA.

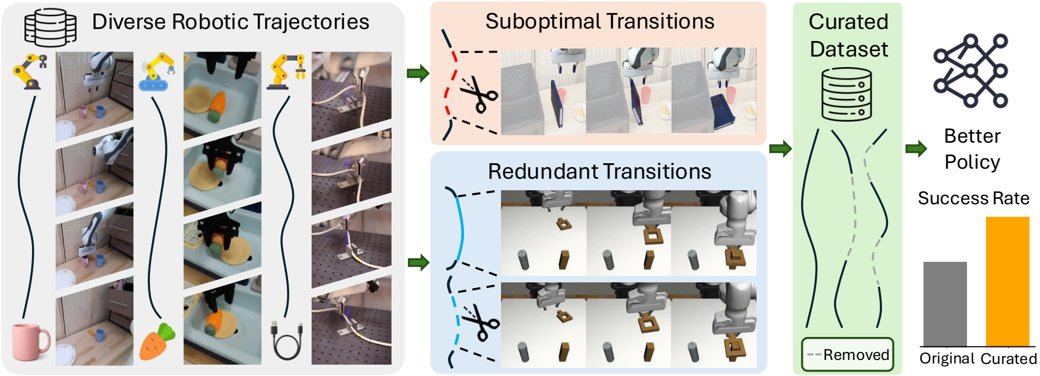


Figure 1: **SCIZOR overview.** Each trajectory from the original robotic datasets is simultaneously passed through the suboptimal transitions removal module and the redundant transitions removal module. Each module removes data based on its own threshold, resulting in a curated dataset. A policy trained on the curated dataset achieves a higher success rate.

to maximize data utilization is *the need for curating at the finest granularity: we must evaluate the quality of individual state-action pairs*. For instance, a trajectory may include an initial failed grasp attempt followed by a successful recovery, containing both suboptimal and valuable segments. Effective curation should isolate and remove only the uninformative or erroneous segments, like the failed grasp, while preserving segments that provide useful learning signals. The different impact of individual data points for learning has also been underscored in prior work, such as weighted behavior cloning [4, 11, 12]. However, current large-scale imitation learning curation methods have yet to address data quality at the transaction level. Existing approaches typically curate by reweighting entire dataset domains [13, 5] or discarding entire trajectories [6, 14], overlooking the contribution and quality of individual state-action pairs.

Effectively curating individual state-action pairs is challenging, as robot demonstrations typically lack dense reward annotations, making it difficult to assess the quality of every interaction step. To address this, we adopt a *self-supervised* approach for filtering low-quality state-action pairs, offering a scalable solution for improving data quality in imitation learning. Our work is motivated by two key observations: (1) *suboptimal* transitions, which contain undesirable actions like collision, jittering, and other erroneous actions, can degrade policy performance by reinforcing incorrect behaviors; and (2) *redundant* transitions, which repeat common patterns excessively, can dilute the learning signal by dominating other informative and diverse samples.

For effective large-scale imitation learning, we introduce SCIZOR, a **s**elf-supervised data **c**uration method that reduces dataset s**iz**e by filtering out sub**o**ptimal and **r**edundant state-action pairs. First, to identify suboptimal data without access to reward information, we train a self-supervised task progress predictor using temporal distance classification [15, 16, 17], and remove frames that do not demonstrate meaningful progress toward the task goal. Second, to remove redundant data, a key insight is that some segments may appear visually similar while differing substantially in the executed actions. Therefore, both visual observations and their corresponding actions must be considered for effective deduplication. To this end, we apply deduplication [7] using joint representations of state and action to identify and filter redundant state-action pairs. We then filter out frames based on similarity scores to reduce repetition while preserving dataset diversity. The suboptimal frame filter targets harmful or noisy supervision, while the redundancy filter removes overrepresented patterns. Together, the two deletion strategies complement each other by targeting distinct modes of low-quality data.

In summary, our key contributions are as follows:

- We propose a unified framework for data curation that filters both suboptimal and redundant state-action pairs in large-scale imitation learning datasets.

- We introduce a suboptimality detector based on self-supervised task progress estimation, and a deduplication module that removes repetitive data to preserve data diversity.
- We empirically demonstrate that SCIZOR improves policy performance across diverse large-scale imitation learning benchmarks, showing on average 15.4% improvement.

2 Related Work

Imitation Learning on Large-Scale Robot Datasets. Imitation learning has been a popular approach to learn robot policy from human demonstrations [18, 19, 1, 20] to scale up robot policy generalization and enable diverse behaviors, there has recently been progress in large-scale multi-task imitation learning [21, 22, 13, 23, 24, 25] trained on robot trajectory data of a wide variety of tasks. This progress is driven not only by advances in policy architectures [21, 1, 26, 27], but more importantly the collection of large-scale datasets in both real-world [28, 29, 30, 31] and simulation [32, 33]. These datasets are often collected from multiple institutions using varied hardware configurations and teleoperation systems [34, 35, 36, 37, 38, 39, 38], resulting in inconsistencies in quality and redundancy across different datasets. Although robotics datasets have been scaled to unprecedented sizes, the study of dataset quality and data curation methods remains preliminary.

Data Curation in Vision and Language Models. Data curation, which is the selection and filtering of data for better training results, have been extensively studied in both computer vision and language modeling to address the challenges posed by large-scale, heterogeneous datasets [8]. In vision, LAION-5B [9] uses pretrained encoders like CLIP to assign data quality on the samples. In language modeling, data mixture strategies like DoReMi [10] balance various data sources for distribution robustness, while deduplication methods like SemDeDup [7] remove near-duplicates using semantic embeddings. Data Filtering Networks [40] trains a neural network to distinguish informative versus less-informative data, while Ask-LLM [41] uses instruction-tuned LLMs to assess the quality of training examples directly. Meanwhile, Less-Is-More-RL [42] shows how pruning suboptimal data can improve downstream policy performance in reinforcement learning settings.

Data Curation for Robotics. Data quality has been known to affect policy learning performance for robotics [1, 43]. There have been studies in improving human demonstration quality, albeit in small-scale tasks, by automatic ranking [2], or eliciting compatible behavior from humans during the data collection process [44]. As progress in general-purpose, large-scale robot learning continues, there has been growing interest in curating large-scale datasets for robot learning [13, 23, 5, 6, 14]. Octo [13] and OpenVLA [23] perform ad-hoc *dataset-level* curation by heuristically tuning a set of weights for data mixtures, balancing the dataset composition; Remix [5] automates this dataset-level curation with distributionally robust optimization. DemInf [6] performs *trajectory-level* curation with mutual information as a trajectory quality estimator, and Demo-SCORE [14] also performs trajectory-level curation, but has to rely on online rollout performance.

3 Self-Supervised Data Curation for Large-Scale Imitation Learning

We introduce our data curation framework, SCIZOR, which performs fine-grained filtering of low-quality data in a self-supervised manner to improve imitation learning policy performance. We begin by introducing key formulations and background, followed by two core components of our method: (1) a self-supervised suboptimal transitions removal module and (2) a similarity-based state-action deduplication module that filters redundant transitions.

3.1 Preliminaries and Formulations

We formulate a robot manipulation task as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p_0, \gamma)$ representing the state space, action space, reward function, transition probability, initial state distribution, and discount factor. Given the current state $s_t \in \mathcal{S}$, the robot action $a_t \in \mathcal{A}$ is drawn from the policy $\pi(\cdot \mid s_t)$. The objective of imitation learning is to learn a policy π_R

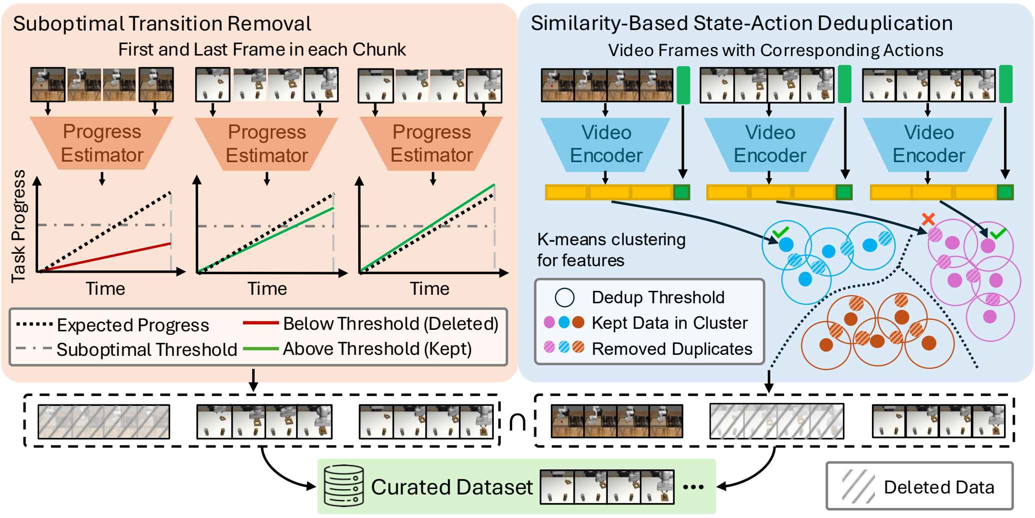


Figure 2: **SCIZOR's architecture.** We apply two curation modules: (1) *Suboptimal transition removal*, where we estimate chunk progress from its first and last frames and discard those below a threshold; (2) *State-action deduplication*, where we encode all frames, cluster their features via K-means, and remove frames whose intra-cluster cosine similarity exceeds a threshold.

parameterized by θ that maximizes the log-likelihood of actions a conditioned on the states s: $\theta^* = \arg\max_{\theta} \mathbb{E}_{(s,a)\sim\mathcal{D}_{expert}}[\log \pi_\theta(a \mid s)]$, where (s, a) are samples from the human demonstration dataset $\mathcal{D}_{expert}$. Our data curation objective is to refine $\mathcal{D}_{expert}$ by filtering out suboptimal or redundant samples to improve downstream policy performance. This is achieved by assigning a quality score to each sample and excluding those below a threshold.

3.2 Suboptimal Transition Removal via Progress Estimation

Human demonstrations often contain both proficient and suboptimal segments in the same trajectory with no explicit signals. Manually labeling suboptimal segments is labor-intensive and not scalable. We propose a self-supervised approach to detect suboptimal behaviors based on the intuition that *progress toward task completion should increase steadily over time*. By training a model to predict task progress between pairs of observations, we learn a proxy for how much progress the agent makes over a given interval. If the predicted progress for a segment is unexpectedly lower than expected progress, it can serve as a signal of suboptimality. This allows us to automatically identify and filter out segments that deviate from making progress, without requiring any manual annotations.

Defining Suboptimality with Task Progress. Inspired by temporal distance classification in self-supervised representation learning [15], we evaluate action quality by estimating the extent of task progress between two timesteps, i and $i + T$. Specifically, we define a progress function $f : S_{i,i+T} \rightarrow T_p$, which inputs a sub-trajectory $S_{i,i+T}$ from timestep i to $i + T$, and predicts the progress T_p made over the sub-trajectory $S_{i,i+T}$ towards completion. *Intuitively, T_p measures the temporal distance that the robot has moved the task forward over the sub-trajectory $S_{i,i+T}$, measured in seconds.* We then compare this predicted progress T_p to the actual elapsed time T. If the predicted progress is significantly lower than the elapsed time, this means the robot is behind schedule (i.e., progressing more slowly than expected), meaning that the sub-trajectory is suboptimal. The suboptimality score for the sub-trajectory $S_{i,i+T}$ is defined as $V_{i,i+T} = T - T_p$.

Predicting Task Progress. Rather than regressing a real-valued progress estimate, we cast progress prediction as classification over discrete temporal bins, which is empirically more robust [15]. We discretize the temporal gap into B bins, where each bin is a time interval in seconds. To predict task progress for a sub-trajectory $S_{i,i+T}$, we train a task progress classifier to classify the bin corresponding to the time T between the start and end states. Empirically, we set $B = 5$, sample each sub-trajectory as 2 seconds, and bins to be $[0, 0.5)$, $[0.5, 1.0)$, $[1.0, 2.0)$, $[2.0, 5.0)$, and $[5.0, +\infty)$.

Assigning Suboptimal Scores to Individual Samples. Although we first compute a suboptimality score at the sub-trajectory level, our ultimate goal is to assign a suboptimality score to every individual transition to enable more fine-grained data filtering. Therefore, we aim to assign a sample-level suboptimal score V_i to each transition at time i, based on the sub-trajectory-level suboptimal scores. We began by computing sub-trajectory-level suboptimal scores, $V_{0,T}, V_{1,1+T} ..., V_{N,N+T}$. Assuming that every sample within a sub-trajectory have equal probability to be suboptimal, each sub-trajectory score should be evenly distributed across its T samples, for a sub-trajectory $S_{i,i+T}$ with $V_{i,i+T}$, it will contribute each transition within the sub-trajectory by $\frac{1}{T}V_{i,i+T}$. Therefore, the aggregated sample-level suboptimality score is then computed as: $\hat{V}_i = \sum_{t=i-T}^{i} \frac{1}{T} V_{t,t+T}$. Further, we apply the temporal discounting to our assigning, which gives us $V_i = \sum_{t=0}^{i} \gamma^{i-t} \hat{V}_t$ where $\gamma \in [0, 1]$ is the discount factor. This captures the influence of past actions on the current state, while prioritizing more recent behavior.

Lastly, we observe that the overall quality of its trajectory can shape the quality of each action; for example, a human operator who is consistently expert-level or error-prone affects the quality of every action in the demonstration he/she collect. We therefore want each transition's score to reflect not only its local quality but also the overall quality of the trajectory it belongs to. To compute the final curation score for each transition, we use a weighted combination of its suboptimality score with the mean score of the transitions across all timesteps, $\frac{1}{N}\sum_{j=1}^{N}(V_j)$. We refer to this combination as the mixture of transition and trajectory scores. Specifically, the final score is computed as $0.5 \times \frac{1}{N}\sum_{j=1}^{N}(V_j) + 0.5 \times V_i$.

Removing Suboptimal Samples. During policy training, we compute the suboptimality score for every sample as described above. During data curation, we define a suboptimal threshold ϵ_s and exclude transitions with suboptimality scores above this threshold from the training process. Note that each sample here preserves an observation history and an action sequence for algorithms that are history-dependent (e.g., BC-Transformer) and that utilize action chunking.

3.3 Similarity-Based State-Action Deduplication

Large-scale imitation learning datasets often include many visually and behaviorally similar sequences, for example, repeated demonstrations of the same skill in nearly identical contexts. Training directly on all such data can hinder policy generalization by overemphasizing common patterns while underrepresenting rare but informative cases. To mitigate this, we introduce a similarity-based deduplication method that filters out redundant data.

A key insight is that some segments may appear visually similar, yet differ in task intent or executed actions. To avoid discarding meaningful variations, effective deduplication must consider both the visual states and actions. To this end, we propose a similarity-based deduplication method that utilizes *joint* representations of visual states and actions to identify and filter redundant state-action pairs.

Defining State-Action Duplicates. Prior work on semantic deduplication [7] has focused on curating large image datasets by removing semantically similar data pairs based solely on visual features. However, such visual-only deduplication methods are not well-suited for sequential decision-making tasks like imitation learning in robotics, where action dynamics play a crucial role. In this work, we extend the idea of semantic deduplication to the imitation learning domain by incorporating both visual states and action information. Specifically, we define state-action duplicates as state-action chunks $(S_{i,i+T}, a_{i,i+T})$ that are visually similar and lead to comparable actions, reflecting redundant patterns that contribute little to learning diversity.

Generating State-Action Features. We first divide the dataset into non-overlapping sub-trajectories, each consisting of a state-action sequence $(S_{i,i+T}, a_{i,i+T})$, where each chunk spans a fixed duration T. Given the variations in recording frequency across datasets, we uniformly subsample $N = 8$ RGB images from each chunk for consistency. As raw visual data is high-dimensional and not directly suitable for similarity computation, we employ the Cosmos video encoder [45], a pre-trained model that encodes both temporal and semantic information from videos, to extract a compact 1D video feature vector z_v. We then concatenate the actions to the visual embedding to form a joint state-action feature z_{v+a}.

Removing Duplicated Samples. We begin by performing K-means clustering to group semantically similar state-action chunks. Within each cluster, we compute pairwise cosine distances among all chunks. For each chunk, its similarity score is defined as the minimum distance it has with any other chunk in the same cluster. We identify as duplicates those chunks whose maximum similarity exceeds a defined threshold, ϵ_d, as they are highly similar to at least one other sample in their cluster. These chunks will be filtered out during policy training with a duplication mask.

3.4 Unified Threshold Across Datasets

When curating data, we remove all samples with a score above either ϵ_s or ϵ_d. To generalize SCIZOR to different datasets, we need to find a unified threshold. We run SCIZOR on the RoboMimic and OXE$_{\text{Magic}}$ datasets under different curation thresholds and find that $\epsilon s = 0.58$ and $\epsilon_d = 0.99$ yield the best performance on both datasets. We then adopt this unified threshold for all experiments. For more details, please refer to Appendix A.

4 Experiments

In our experiments, we aim to address the following questions: 1) How much does SCIZOR improve imitation learning policy performance? 2) What advantage does SCIZOR's fine-grained state-action curation offer over trajectory- or dataset-level curation in prior work? 3) What design components contribute most to SCIZOR? 4) What types of low-quality samples can SCIZOR identify and remove?

4.1 Experimental Setup

Datasets and Training Details. We evaluate our method on three robotic benchmarks for imitation learning, chosen to represent a range of real-world scenarios: a large-scale crowdsourced dataset, a dataset featuring varying levels of human expertise, and a human-in-the-loop dataset with mixed data distributions. This selection enables us to evaluate SCIZOR's effectiveness across various scenarios and diverse data regimes. For full dataset details, see Appendix B.1.

- **Open-X-Embodiment (OXE) [29]**: A large-scale collection of over one million real-world robotic trajectories. We use the Simpler environment [46] and benchmark on two tasks: *Pick Can* and *Move Near*. We train the Octo model [13] with two random seeds and use the same "Magic Soup" weighting. This setting evaluates SCIZOR's scalability to large and diverse datasets.
- **RoboMimic [1]**: A dataset and benchmark containing human-collected trajectories of varying proficiency. We use the simulated Multi-Human dataset for the *Can* and *Square* tasks to be consistent with the baseline comparison. We train the BC policy provided in the benchmark with three random seeds. This setting evaluates SCIZOR's ability to curate demonstrations of mixed quality.
- **Sirius-Fleet [47]**: A real-world multi-task dataset comprising 1,500 policy rollouts with human interventions. Our real-world evaluation spans four task sets comprising eight tasks. We train the BC-Transformer policy used in the paper with three random seeds. This setting evaluates SCIZOR's ability to curate mixed data from both autonomous policies and human corrections.

Baselines. We benchmark SCIZOR against 3 baselines, each highlighting a different aspect of data curation. We compare with **Uniform** to show the effectiveness of SCIZOR, with **DemoInf** and **Re-Mix** to show that fine-grained curation offers advantages over coarser filtering strategies.

- **Uniform**: A baseline that uniformly deletes the same percentage of data as other methods to control for dataset size. This comparison ensures that the improvement observed with SCIZOR is attributed to *which* specific samples are removed, not simply to the reduced dataset size itself.
- **DemoInf [6]**: A *trajectory-level* method that estimates mutual information between states and actions for each trajectory as a quality score and removes low-quality trajectories with insufficient

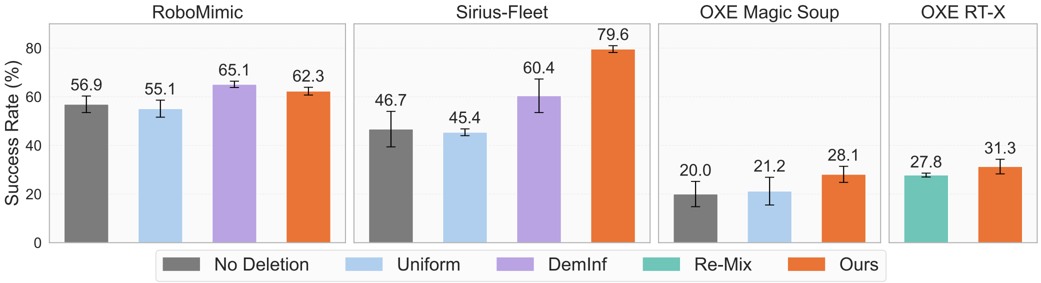


Figure 3: **Performance comparison across different datasets.** We use the unified threshold for SCIZOR and report success rates on 4 datasets. We found that SCIZOR achieves the strongest performance and outperforms the baselines.

contribution. We apply it to the RoboMimic and the Sirius-Fleet dataset, following the same dataset settings as in the original paper. We delete the same percentage of data as other methods.

- **Re-Mix [5]**: A *dataset-level* method that learns data mixture weights for the "RT-X" variant of the OXE datasets. To ensure consistency, we train the Octo-small model on $\text{OXE}_{\text{RT-X}}$ for SCIZOR, while directly adopting their learned weights for Re-Mix.

4.2 Experimental Results

RQ1: How much does SCIZOR improve imitation learning policy performance? Figure 3 summarizes SCIZOR's impact on policy success rates across all three benchmarks. Compared to training on the full dataset, SCIZOR delivers absolute gains of 5.4% on RoboMimic, 8.1% on $\text{OXE}_{\text{Magic}}$, and 32.9% on the Sirius-Fleet real-robot tasks. It also surpasses uniform curation by 16.1% on average, indicating that SCIZOR has a targeted selection of samples to be deleted. These improvements demonstrate that SCIZOR's data curation consistently filters out low-quality samples and improves policy learning in both simulated and real-world robotic environments.

RQ2: What advantage does SCIZOR's fine-grained state-action curation offer over trajectory- or dataset-level curation in prior work? To validate the effectiveness of fine-grained curation on state-action pairs, we compare SCIZOR with two baseline methods: a trajectory-level curation method, Deminf [6], and a dataset-level curation method, Re-Mix [10]. Deminf estimates the average contribution of a trajectory towards the mutual information between states and actions in the entire dataset. Re-Mix treats each subset of data as a different "domain" and uses a distributionally robust optimization technique to assign weights to sub-datasets. To ensure a fair comparison, we apply SCIZOR to the same RT-X mixture setting used by Re-Mix. As shown in Figure 3 SCIZOR outperforms Re-Mix by 3.5% on average. In the RoboMimic dataset, SCIZOR has not outperformed DemInf, as the dataset is explicitly divided into three levels of trajectory quality, making trajectory-level filtering particularly effective. In contrast, SCIZOR significantly outperforms DemInf by 19.2% on the Sirius-Fleet dataset, where the mixed sources of policy and human actions result in uneven data quality distribution. This suggests that fine-grained state-action curation may be beneficial in datasets with complex and uneven quality distributions.

Table 1: **Ablation studies**: Performance comparison across three datasets (RoboMimic, Sirius-Fleet, and OXE). Our approach consistently outperforms partial ablations, highlighting the importance of combining both components.

	RoboMimic	**Sirius-Fleet**	**$\text{OXE}_{\text{Magic}}$**
Suboptimal-Removal Only	60.9 ± 1.8	64.2 ± 2.6	25.3 ± 2.9
Deduplication Only	48.3 ± 0.8	63.3 ± 6.9	22.1 ± 0.9
SCIZOR (Ours)	**62.3 ± 1.6**	**79.6 ± 1.4**	**28.1 ± 3.3**

RQ3: What design components contribute most to SCIZOR? We first ablate the suboptimal data removal and the deduplication in Table 1. We run the experiments only removing suboptimal data

and duplicated data, and remove the same amount of data in each dataset as SCIZOR. We find that both suboptimal removal and deduplication individually lead to improvements over the baseline, but neither alone is sufficient to match the full performance of SCIZOR. Suboptimal removal is generally more effective than deduplication, but combining both components leads to the largest gains across all datasets.

Table 2: **Variations of SCIZOR's suboptimal data strategies**: We evaluate different scoring strategies for suboptimal data removal: (i) without mixture of transition and trajectory scores, (ii) without temporal discounting, and (iii) the full proposed method (Ours). Results are reported across four tasks, showing that the full version consistently outperforms the alternatives.

	RoboMimic Can	**RoboMimic Square**	**$\text{OXE}_{\text{RT-1}}$ Pick**	**$\text{OXE}_{\text{RT-1}}$ Move**
SCIZOR w/o mixture	81.3 ± 0.6	36.0 ± 1.4	21.8 ± 7.9	12.4 ± 4.6
SCIZOR w/o discount	79.6 ± 1.4	31.5 ± 5.5	20.7 ± 6.4	9.4 ± 1.4
SCIZOR (Ours)	**87.3 ± 0.7**	**37.2 ± 2.5**	**30.9 ± 8.4**	**17.5 ± 1.0**

We further investigate SCIZOR's scoring strategy for suboptimal data classifier in Table 2 by ablating two key components: (i) the transition–trajectory score mixture and (ii) temporal discounting discussed in Section 3.2. We train Octo on the OXE "RT-1" variant [21] with three seeds for faster iteration. Omitting either component consistently degrades performance across all four tasks, highlighting their importance. Temporal discounting lets SCIZOR propagate evidence of suboptimality backward in time, so that transitions leading to poorer future states can also be identified in addition to directly poor actions. The mixture of transition-level and trajectory-level scores balances these fine-grained penalties with an overall assessment of each demonstration's quality, making it easier to filter out inherently low-quality data (for example, trajectories recorded by non-expert operators). Together, these mechanisms yield the strongest gains in suboptimal data removal.

RQ4: What types of low-quality samples can SCIZOR identify and curate? To qualitatively visualize the suboptimal data identified, we examine the low-quality data classified and investigate the types of low-quality data they represent. From each of the RoboMimic and Sirius-Fleet datasets, we randomly select 100 demonstrations flagged with at least one suboptimal segment. We then manually visualize and classify every suboptimal segment across these trials to generate the pie chart. Figure 4 illustrates the distribution of suboptimal transitions identified by SCIZOR. *Manipulation Failure* refers to errors during grasping — e.g., failed grasps, accidental drop of objects. *Pause* denotes transitions with very little or no movement. *Stuck at Collision* describes cases where the gripper or held object collides, leading to a halt. *Slow* captures motion that proceeds noticeably below the normal task speed. *Move Back and Forth* indicates aimless motions that do not contribute to task progress. *False Positive* labels misclassified or ambiguous transitions. The most significant fractions are *Slow Motion*, *Manipulation Failure*, and *Pause*, indicating that the task-progress classifier identifies semantically meaningful errors rather than spurious noise. Appendix B.4 visualizes representative examples.

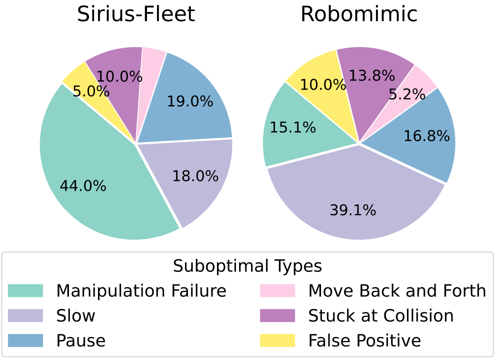


Figure 4: **Breakdown of suboptimal types classified by SCIZOR.** The three dominant failure modes predicted by SCIZOR's suboptimal classifier are *Slow Motion*, *Manipulation Failure*, and *Pause*, showing SCIZOR removes semantically meaningful transitions.

5 Conclusion

We introduce SCIZOR, a self-supervised data curation method that filters suboptimal and redundant state-action pairs to improve imitation learning performance. It combines a task progress predictor to remove suboptimal frames with a similarity-based deduplication module to eliminate overrepresented patterns. By curating the dataset, SCIZOR consistently enhances policy performance across diverse imitation learning benchmarks and outperforms other data curation approaches on large datasets. Future work could explore more adaptive thresholding strategies to achieve optimal deletion ratios and improve the representation of state-action pairs for better curation performance.

6 Limitation

While SCIZOR improves policy success in imitation learning, it has several limitations, which we discuss in detail below:

Curation Threshold: The deletion threshold in SCIZOR is currently determined empirically. Future work could derive a theoretical framework to identify this threshold more systematically.

Deduplication Representation: SCIZOR's deduplication module currently concatenates action and state features. While it performs well in our experiments, future work could explore more expressive or learned representations [48, 49] that better integrate the action and state spaces.

Dependence on Demonstration Quality: SCIZOR assumes that most demonstrations within a trajectory are of good quality, as we rely on self-supervised learning to learn from the majority of the data. If poor-quality demonstrations dominate, the method may become less effective. Future work could focus on better leveraging low-quality data by identifying and utilizing useful segments.

Linear Task Progress Assumption: SCIZOR assumes linear task progression without slow or repetitive behaviors. However, real-world tasks, like stirring food repeatedly or waiting for it to cook, often involve such behaviors. Future work could adapt the method to better handle these behaviors.

References

[1] 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, 2021. URL https://arxiv.org/abs/2108.03298.

[2] D. S. Brown, W. Goo, and S. Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations, 2019. URL https://arxiv.org/abs/1907.03976.

[3] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation, 2025. URL https://arxiv.org/abs/2410.18647.

[4] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

[5] J. Hejna, C. Bhateja, Y. Jian, K. Pertsch, and D. Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

[6] J. Hejna, S. Mirchandani, A. Balakrishna, A. Xie, A. Wahid, J. Tompson, P. Sanketi, D. Shah, C. Devin, and D. Sadigh. Robot data curation with mutual information estimators. 2025. URL https://arxiv.org/abs/2502.08623.

[7] A. Abbas, K. Tirumala, D. Simig, S. Ganguli, and A. S. Morcos. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.

[8] A. Albalak, Y. Elazar, S. M. Xie, S. Longpre, N. Lambert, X. Wang, N. Muennighoff, B. Hou, L. Pan, H. Jeong, C. Raffel, S. Chang, T. Hashimoto, and W. Y. Wang. A survey on data selection for language models, 2024. URL https://arxiv.org/abs/2402.16827.

[9] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. Laion-5b: An open large-scale dataset for training

next generation image-text models. *ArXiv*, abs/2210.08402, 2022. URL https://api.semanticscholar.org/CorpusID:252917726.

[10] S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. Liang, Q. V. Le, T. Ma, and A. W. Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *ArXiv*, abs/2305.10429, 2023. URL https://api.semanticscholar.org/CorpusID:258741043.

[11] Z. Wang, A. Novikov, K. Zolna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, and N. de Freitas. Critic regularized regression. volume 33, pages 7768–7778, 2020.

[12] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *ICLR*, 2021.

[13] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

[14] A. S. Chen, A. M. Lessing, Y. Liu, and C. Finn. Curating demonstrations using online experience, 2025. URL https://arxiv.org/abs/2503.03707.

[15] Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018.

[16] P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning, 2017. URL https://arxiv.org/abs/1612.06699.

[17] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning, 2021. URL https://arxiv.org/abs/2106.03911.

[18] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NeurIPS*, 1989.

[19] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018.

[20] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *CoRL*, 2021.

[21] 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. J. 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. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[22] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[23] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[24] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[25] NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

[26] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.

[27] 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, 2024.

[28] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.

[29] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[30] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, et al. Droid: A large-scale in-the-wild robot manipulation dataset, 2024.

[31] AgiBot-World-Contributors, Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Huang, S. Jiang, Y. Jiang, C. Jing, H. Li, J. Li, C. Liu, Y. Liu, Y. Lu, J. Luo, P. Luo, Y. Mu, Y. Niu, Y. Pan, J. Pang, Y. Qiao, G. Ren, C. Ruan, J. Shan, Y. Shen, C. Shi, M. Shi, M. Shi, C. Sima, J. Song, H. Wang, W. Wang, D. Wei, C. Xie, G. Xu, J. Yan, C. Yang, L. Yang, S. Yang, M. Yao, J. Zeng, C. Zhang, Q. Zhang, B. Zhao, C. Zhao, J. Zhao, and J. Zhu. AgiBot World Colosseo: A Large-scale Manipulation Platform for Scalable and Intelligent Embodied Systems. *arXiv preprint arXiv:2503.06669*, 2025.

[32] 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. In *7th Annual Conference on Robot Learning*, 2023.

[33] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[34] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale, 2024. URL https://arxiv.org/abs/2308.12952.

[35] A. . Team, J. Aldaco, T. Armstrong, R. Baruch, J. Bingham, S. Chan, K. Draper, D. Dwibedi, C. Finn, P. Florence, S. Goodrich, W. Gramlich, T. Hage, A. Herzog, J. Hoech, T. Nguyen, I. Storz, B. Tabanpour, L. Takayama, J. Tompson, A. Wahid, T. Wahrburg, S. Xu, S. Yaroshenko, K. Zakka, and T. Z. Zhao. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation, 2024.

[36] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024.

[37] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[38] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[39] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik. Learning visuotactile skills with two multifingered hands. *arXiv preprint arXiv:2404.16823*, 2024.

[40] A. Fang, A. M. Jose, A. Jain, L. Schmidt, A. Toshev, and V. Shankar. Data filtering networks, 2023. URL https://arxiv.org/abs/2309.17425.

[41] N. Sachdeva, B. Coleman, W.-C. Kang, J. Ni, L. Hong, E. H. Chi, J. Caverlee, J. McAuley, and D. Z. Cheng. How to train data-efficient llms, 2024. URL https://arxiv.org/abs/2402.09668.

[42] X. Li, H. Zou, and P. Liu. Limr: Less is more for rl scaling, 2025. URL https://arxiv.org/abs/2502.11886.

[43] S. Belkhale, Y. Cui, and D. Sadigh. Data quality in imitation learning, 2023. URL https://arxiv.org/abs/2306.02437.

[44] K. Gandhi, S. Karamcheti, M. Liao, and D. Sadigh. Eliciting compatible demonstrations for multi-human imitation learning. In *Conference on Robot Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:252918784.

[45] NVIDIA, N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding, D. Dworakowski, J. Fan, M. Fenzi, F. Ferroni, S. Fidler, D. Fox, S. Ge, Y. Ge, J. Gu, S. Gururani, E. He, J. Huang, J. Huffman, P. Jannaty, J. Jin, S. W. Kim, G. Klár, G. Lam, S. Lan, L. Leal-Taixe, A. Li, Z. Li, C.-H. Lin, T.-Y. Lin, H. Ling, M.-Y. Liu, X. Liu, A. Luo, Q. Ma, H. Mao, K. Mo, A. Mousavian, S. Nah, S. Niverty, D. Page, D. Paschalidou, Z. Patel, L. Pavao, M. Ramezanali, F. Reda, X. Ren, V. R. N. Sabavat, E. Schmerling, S. Shi, B. Stefaniak, S. Tang, L. Tchapmi, P. Tredak, W.-C. Tseng, J. Varghese, H. Wang, H. Wang, H. Wang, T.-C. Wang, F. Wei, X. Wei, J. Z. Wu, J. Xu, W. Yang, L. Yen-Chen, X. Zeng, Y. Zeng, J. Zhang, Q. Zhang, Y. Zhang, Q. Zhao, and A. Zolkowski. Cosmos world foundation model platform for physical ai, 2025. URL https://arxiv.org/abs/2501.03575.

[46] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[47] H. Liu, Y. Zhang, V. Betala, E. Zhang, J. Liu, C. Ding, and Y. Zhu. Multi-task interactive robot fleet learning with visual world models, 2024. URL https://arxiv.org/abs/2410.22689.

[48] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv: 2501.09747*, 2025.

[49] S. Li, Y. Gao, D. Sadigh, and S. Song. Unified video action model. *arXiv preprint arXiv: 2503.00200*, 2025.

[50] R. Shah, R. Martín-Martín, and Y. Zhu. Mutex: Learning unified policies from multimodal task specifications. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=PwqiqaaEzJ.

[51] K. Pertsch. Rlds dataset modification, 2024. URL https://github.com/kpertsch/rlds_dataset_mod. GitHub repository.

A Method details

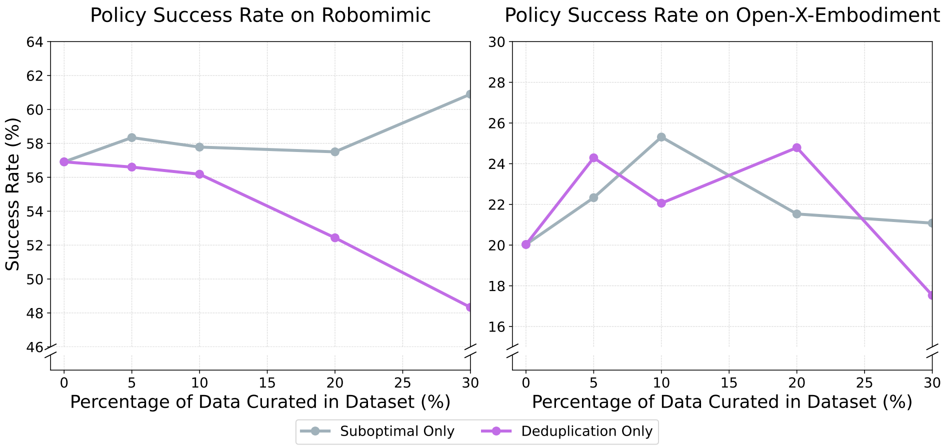


Figure 5: The performance of the Suboptimal-Only method and the Deduplication-Only method when deleting different percentages of data on the Robomimic and $\text{OXE}_{\text{Magic}}$ dataset.

A.1 Determining the Unified Threshold

To find a unified threshold for both suboptimal-transition removal and similarity-based state–action deduplication, we run SCIZOR with only one sub-method at a time on the RoboMimic and $\text{OXE}_{\text{Magic}}$ datasets with deletion ratios of 10%, 20%, and 30%.

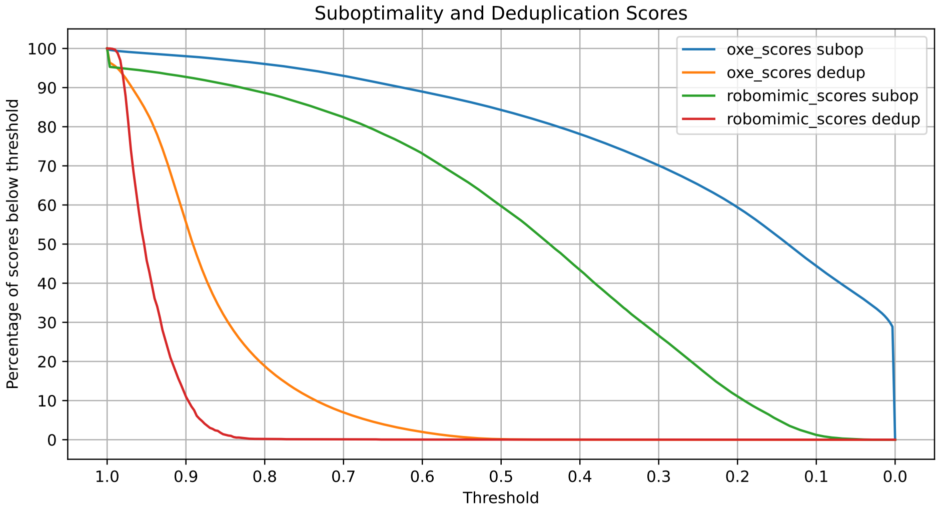


Figure 6: Deletion ratio as a function of the chosen threshold for suboptimal-transition removal and state–action deduplication on the Robomimic and $\text{OXE}_{\text{Magic}}$ datasets.

Figure 5 shows that Suboptimal-Only achieves its highest success rate at 30% deletion on RoboMimic and 10% deletion on OXE. Deduplication-Only performs best with 0% deletion on RoboMimic and 20% deletion on OXE.

Next, we plot the deletion ratio as a function of the threshold in Figure 6. We observe that: A suboptimal threshold of 0.58 yields deletion rates of 29.5% on RoboMimic and 11.9% on OXE, closely matching their respective optimal ratios. A deduplication threshold of 0.99 results in only 0.3%

deletion on RoboMimic—insufficient to harm performance—and 4.5% deletion on OXE, which still provides a notable improvement, very close to deleting 20%.

Finally, we apply the unified threshold to all the datasets, and yield the deletion ratio list in Table 3.

Table 3: **Deletion Ratio** of all the datasets when unified threshold applied

	RoboMimic	**Sirius-Fleet**	**OXE$_{\text{Magic}}$**	**OXE$_{\text{RT-X}}$**	**OXE$_{\text{RT-1}}$**
Suboptimal-Only	29.3%	4.7%	11.9%	15.0%	9.2%
Deduplication-Only	0.3%	3.2%	4.5%	0.8%	0.5%
SCIZOR (Total)	29.6%	7.9%	15.8%	15.8%	9.7%

A.2 Task Progress Prediction Model Architecture.

Given a sub-trajectory $S_{i,i+T}$, the model takes image observation at timesteps i and $i+T$ as inputs. These observations are independently encoded using a frozen DINO-V2 model to obtain the visual features, which provide robust and generalizable visual representations thanks to its self-supervised pretraining on diverse natural images. We compute the difference between the two feature vectors to obtain a delta feature vector, which emphasizes task-relevant changes and accelerates convergence while discarding redundant static information and is concatenated with a CLS token. The feature vector is then processed through a series of multi-layer self-attention transformer blocks. The output CLS token is then fed into a classification head to produce the predicted progress bin.

A.3 Training the Task Progress Predictor

We divide each trajectory into five equal time bins. For each training example, we randomly select one bin and sample a time interval Δt uniformly within its bounds. The model takes as input the frame at time t and the frame at $t + \Delta t$, and is trained to predict the index of the chosen time bin.

A.4 Fixed Time Duration

All experiments use a constant interval of 2s for progress prediction and state–action feature extraction. Since datasets differ in control frequency, this 2s window corresponds to a dataset-dependent number of transitions per second.

B Experimental Details

B.1 Dataset Details

RoboMimic [1] is a robotic imitation learning dataset and benchmark. It provides trajectories collected by proficient human (PH) or mixed-proficient human (MH) demonstrators. The PH dataset consists of 200 trajectories from a single experienced demonstrator, while the MH dataset includes 300 trajectories from six demonstrators — two "better", two "okay", and two "worse". For our experiment, we use the MH dataset for the "Can" and "Square" tasks.

Sirius-Fleet [47] uses a visual world model to predict sub-optimal behaviors during policy rollout and requests human intervention when needed. The **Sirius-Fleet** dataset is collected over three rounds by allowing the policy to roll out and incorporating human-corrected data for retraining. We utilize the real-world **Sirius-Fleet** dataset, which adopts the Mutex settings [50] and includes 1,500 trajectories. Our real-world evaluation spans four task sets comprising eight tasks.

Open-X-Embodiment (OXE) [29] is a large-scale collection of over one million real-world robotic trajectories. The dataset is multi-task and cross-embodiment, covering various action and observation spaces. We use the RLDS Dataset Modification [51] to unify the action space to 7 DoFs. We employ three variations of the OXE dataset, each selecting different subsets of the original data and applying different weightings: the "Magic Soup" mixture (OXE$_{\text{Magic}}$) used in the Octo paper [13], the “RT-X” mixture ($\text{OXE}_{\text{RT-X}}$) used in the Re-Mix paper [10], and the “RT-1” dataset ($\text{OXE}_{\text{RT-1}}$) from the RT-1 paper [21].

B.2 Training and Evaluation Details

Table 4: Hyperparameter configurations and architectural details for the Robomimic, Sirius-fleet, and OXE datasets used in our experiments.

	RoboMimic	**Sirius-Fleet Real**	$\textbf{OXE}_{\textbf{Magic}}$	$\textbf{OXE}_{\textbf{RT-X}}$	$\textbf{OXE}_{\textbf{RT-1}}$
Architecture	BC	BC-Transformer-GMM	Octo	Octo	Octo
Learning Rate	1e-4	1e-4	3e-4	3e-4	3e-4
Weight Decay	0.1	0.1	0.1	0.1	0.1
Batch Size	16	16	2048	2048	256
Params	23M	35M	93M	93M	93M
Steps	300K	1M	300K	300K	200K
Action Chunk	1	10	4	4	4
Obs History	1	10	2	2	2
GPU	1 L40S 48GB	1 L40S 48GB	32 H100 80GB	32 H100 80GB	8 L40 48GB

We evaluate SCIZOR across various architectures and datasets to demonstrate its applicability to different imitation learning algorithms. We intentionally leave all model architectures and hyperparameters unchanged from the public reference implementations of each dataset, demonstrating that SCIZOR is *plug-and-play*. For the **RoboMimic** and **Sirius-Fleet** experiments, we train each model using three random seeds. For the larger **OXE** experiments, we train the Octo model using two seeds.

On the **RoboMimic** dataset, we train a basic Behavior Cloning (BC) model with MLP layers [1] for 600 epochs. We evaluate every 20 epochs, select the top three checkpoints per random seed, and report the mean success rate over 80 trials per task for each checkpoint, then average across seeds.

For the **Sirius-Fleet** real-robot experiments, we use a BC-Transformer model with a GMM head [1, 4, 47], and train for 2000 epochs. Evaluation is performed at the 2000 epoch, and we report the average success rate of the top three checkpoints for each seed. We run 10 trials per task per seed for this setting.

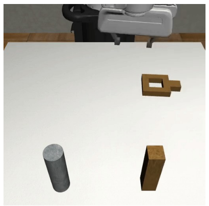
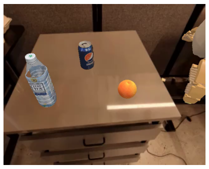
We train the Octo-Small model on all three variations of the **OXE** dataset. For $\textbf{OXE}_{\textbf{Magic}}$ and $\textbf{OXE}_{\textbf{RT-X}}$, training is performed for 300K steps with a batch size of 2048 using two seeds. For $\textbf{OXE}_{\textbf{RT-1}}$, we train for 200K steps with a batch size of 256 using three seeds. Evaluation is conducted in the SIMPLER simulation environment [46] on the “Pick Coke Can” and “Move Near” tasks. We only evaluate on the Visual-Matching setting of SIMPLER, which means there won’t be lighting or texture variation. For each seed, we identify the highest success rate among the last three saved checkpoints, and then average these best performances across seeds. We evaluate 300 trials per checkpoint on “Pick Coke Can” and 240 trials per checkpoint on “Move Near”.

Table 4 provides the detailed hyperparameters and model architectures used in our experiments.

B.3 Evaluation Task Details for each Dataset

Table 5 presents the detailed descriptions and visualizations of the tasks used in our experimental settings. These tasks span both simulated and real-world environments, covering a diverse range of manipulation challenges.

Table 5: Task Name and Description for each setting.

Visualization	Task Name and Description
	Robomimic Can The robot is required to place a coke can from a large bin into a smaller target bin.
	**Robomimic Square** The robot is required to pick a square nut and place it on a rod. Substantially more difficult than Pick Place Can due to the precision required.
	SIMPLER Pick Coke Can The robot is instructed to grasp the empty coke can on the table and lift it up.
	**SIMPLER Move Near** The robot is instructed to move one object next to another object, while the third object serves as a distractor.
	Mutex Mug to Basket The robot is instructed to pick up the blue mug and then place it in the basket.
	Mutex Bowl to Basket The robot is instructed to pick up the red bowl and then place it in the basket.
	Mutex Mug to Oven Tray The robot is instructed to pick up the blue mug and then place it on the oven tray.

continued on next page

continued from previous page

Visualization	Task Name and Description
	Mutex Bread to Plate The robot is instructed to pick up the bread and then place it on the white plate.
	Mutex Mug to Plate The robot is instructed to pick up the pink mug and then place it on the white plate.
	Mutex Bowl to Plate The robot is instructed to pick up the bowl with hot dogs and then place it on the white plate.
	Mutex Book to Caddy The robot is instructed to pick up the book and then place it in the back compartment of the caddy.
	Mutex Cup to Caddy The robot is instructed to pick up the red cup and then place it in the front compartment of the caddy.

B.4 Extra results and Visualization

Table 6, 7, 8 presents the detailed results of different methods on each task of the reported datasets.

Table 6: Success rates on Robomimic across different tasks and methods

	Can	**Square**
Suboptimal-Removal Only	84.0%	37.8%
Deduplication Only	74.3%	22.4%
Random Deletion	78.0%	32.2%
DemInf	**88.9%**	**41.4%**
Scizor (Ours)	84.0%	**40.8%**

Table 7: Success rates on Sirius-Fleet across different tasks and methods

	Book→Caddy	Cup→Caddy	Bowl→Plate	Mug→Plate
No Deletion	40.0%	65.0%	65.0%	35.0%
Suboptimal-Removal Only	65.0%	**75.0%**	70.0%	60.0%
Deduplication Only	45.0%	**75.0%**	80.0%	70.0%
Random Deletion	50.0%	65.0%	55.0%	30.0%
Deminf	**66.7%**	53.3%	60.0%	50.0%
SCIZOR (Ours)	**66.7%**	73.3%	**93.3%**	**83.3%**
	Mug→Basket	Bowl→Basket	Mug→Tray	Bread→Plate
No Deletion	35.0%	60.0%	35.0%	50.0%
Suboptimal-Removal Only	35.0%	90.0%	50.0%	60.0%
Deduplication Only	65.0%	70.0%	60.0%	60.0%
Random Deletion	25.0%	70.0%	30.0%	35.0%
Deminf	50.0%	76.7%	53.3%	63.3%
SCIZOR (Ours)	**66.7%**	**96.7%**	**66.7%**	**90.0%**

Table 8: Success rates on OXE across different tasks, mixtures and methods

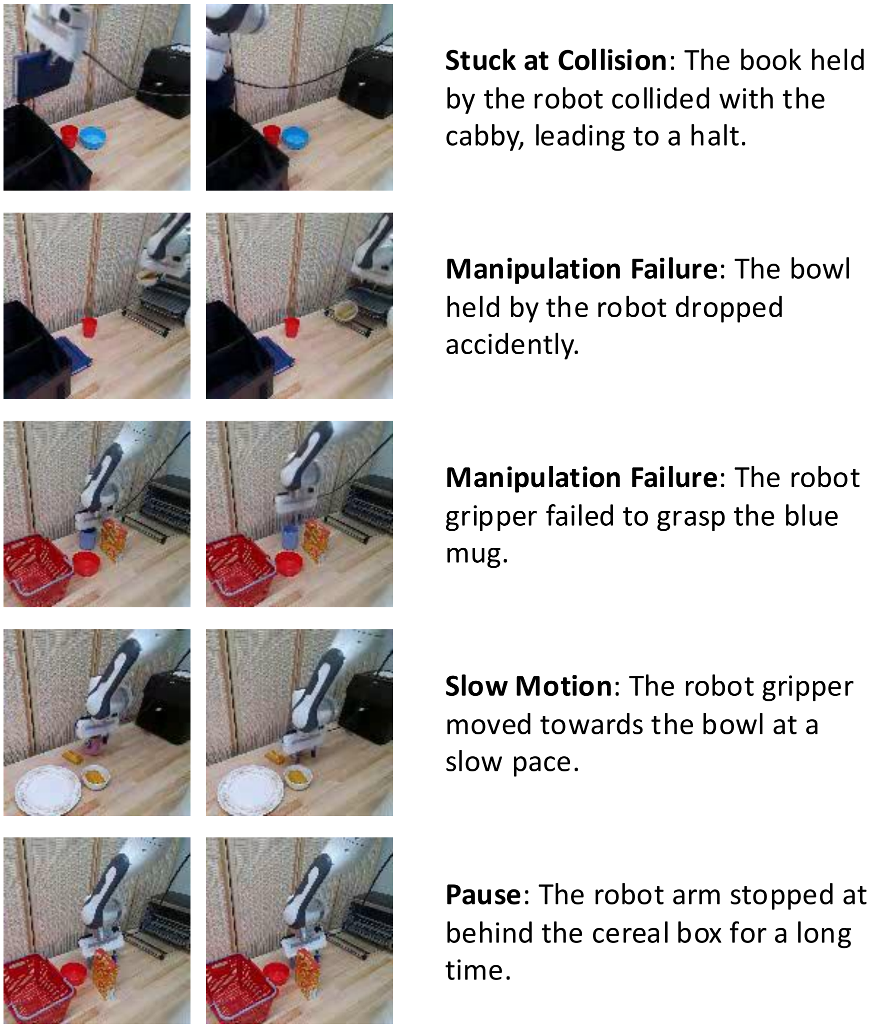
	Pick Can	**Move Near**
$\text{OXE}_{\text{Magic}}$ Mixture		
No Deletion	27.0%	13.1%
Random Deletion	29.1%	12.1%
Suboptimal-Removal Only	33.7%	**16.9%**
Deduplication Only	31.8%	12.3%
SCIZOR (Ours)	**39.5%**	**16.7%**
$\text{OXE}_{\text{RT-X}}$ Mixture		
Remix	40.5%	15.0%
SCIZOR(Ours)	**43.3%**	**19.2%**

We also visualize the suboptimal scenarios identified by SCIZOR across both the Robomimic and Sirius-Fleet datasets in Figure 7, 8. These visualizations highlight common suboptimal modes detected by our method, offering insight into SCIZOR's capabilities.

B.5 Findings during Evaluation

During evaluation of SCIZOR on the Sirius-Fleet dataset, we observed that several failure modes present in the policy trained on the full dataset disappeared when using SCIZOR to curate the dataset.

For example, in the book-to-caddy task, the baseline policy often allowed the book to collide with the caddy, whereas our policy reliably avoids any contact. Furthermore, our policy is noticeably more reactive: when it does encounter a failure, the baseline tends to pause or oscillate in place, but our policy quickly returns to its original trajectory and retries until the task is completed.



Stuck at Collision: The book held by the robot collided with the cabby, leading to a halt.

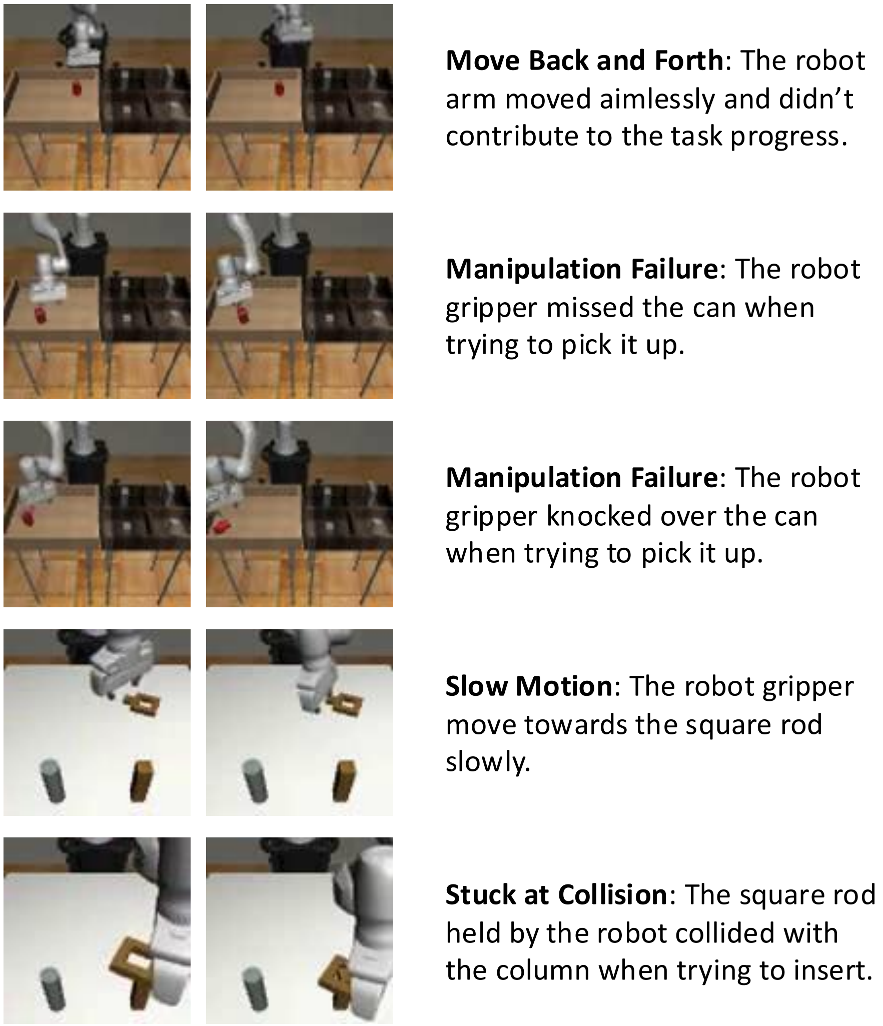
Manipulation Failure: The bowl held by the robot dropped accidently.

Manipulation Failure: The robot gripper failed to grasp the blue mug.

Slow Motion: The robot gripper moved towards the bowl at a slow pace.

Pause: The robot arm stopped at behind the cereal box for a long time.

Figure 7: Visualizations for suboptimal scenarios detected in Robomimic dataset



Move Back and Forth: The robot arm moved aimlessly and didn’t contribute to the task progress.

Manipulation Failure: The robot gripper missed the can when trying to pick it up.

Manipulation Failure: The robot gripper knocked over the can when trying to pick it up.

Slow Motion: The robot gripper move towards the square rod slowly.

Stuck at Collision: The square rod held by the robot collided with the column when trying to insert.

Figure 8: Visualizations for suboptimal scenarios detected in Sirius-Fleet dataset