DataMIL: Selecting Data for Robot Imitation Learning with Datamodels

Shivin Dass[*]
UT Austin

Alaa Khaddaj[*]
MIT

Logan Engstrom
MIT

Aleksander Mądry
MIT

Andrew Ilyas[†]
Stanford University

Roberto Martín-Martín[†]
UT Austin

Abstract: Recently, the robotics community has amassed ever larger and more diverse datasets to train generalist robot policies. However, while these policies achieve strong mean performance across a variety of tasks, they often underperform on individual, specialized tasks and require further tuning on newly acquired task-specific data. Combining task-specific data with carefully curated subsets of large prior datasets *can* produce better specialized policies, but selecting data naively may actually harm downstream performance. To address this, we introduce DataMIL, a data selection framework built on the datamodels paradigm that reasons about data selection in an end-to-end manner, using the policy itself to identify which data points will most improve performance. Unlike standard practices that filter data using human notions of quality (e.g., semantic or visual similarity), DataMIL directly optimizes data selection for task success, allowing us to select data that enhance the policy while dropping data that degrade it. To avoid performing expensive rollouts in the environment during selection, we use a surrogate loss function on task-specific data, allowing us to use DataMIL in the real world without degrading performance. We validate our approach on a suite of 60+ simulation and real-world manipulation tasks—most notably showing successful data selection from the Open X-Embodiment datasets. Our results underscore the importance of end-to-end, performance-aware data selection for unlocking the potential of large prior datasets in robotics. More information at our website.

Keywords: Imitation Learning, Data Curation, Large Robot Datasets

1 Introduction

Recently we have witnessed a revolution in robot learning: inspired by the successes of large-scale language and vision models, the robotics community has begun training large foundation policies [1–10] by amassing large diverse robotic datasets comprising of a variety of tasks, scenes, and robots [3, 11–14]. The resulting generalist policies achieve a strong mean performance across tasks and environments, but often underperform on individual tasks [5, 15], highlighting a gap between generalization and task-specific competence. To bridge this gap, researchers have explored a post-training paradigm [6], where pre-trained foundation models are fine-tuned [5, 6, 15–17] for specific tasks, though this process demands a considerable number of newly acquired task-specific demonstrations. As datasets grow increasingly large and diverse, a natural question arises: *how can we identify and select data from within existing datasets to boost task performance?*

Selecting data to train a high-performing model is a complex endeavor. Naively, it would require testing **each subset** of the data by retraining and evaluating the performance of the trained model. This is expensive for any sizable dataset, becoming infeasible in robotics, where evaluation involves policy rollouts in the real world—a time-consuming and often dangerous procedure. Prior data selection methods in robotics remove the dependency on policy rollouts by selecting data based on heuristics, i.e., assuming that the most useful data is the most similar in language description [18], visually [19], in motion [20], or in state-action pairs [21] to a small number of task-specific demonstrations. While intuitive (and effective in some cases), these heuristics often make several assumptions and fail to consider the real impact of a datapoint on policy performance (see Figure 1 (left)).

In other fields such as natural language processing (NLP) and computer vision (CV), researchers have developed an efficient framework for data selection based on model performance: **datamodels** [22]. Training a datamodel is a meta-process in which the original learning algorithm and model are viewed as a "black box" that consumes data, and the goal is to train an estimator of the black box's behavior as a function of the input data. By avoiding selecting based on heuristics, datamodels stay close to the true optimization objective (trained model performance) while critically reducing the amount of training and evaluation procedures necessary with the original learning algorithm and model to an acceptable level for NLP and CV. However, these evaluations are still unfeasible for policy learning due to the need for real-world rollouts, impeding their application to robotics.

In this work, we introduce **DataMIL** (Datamodels for Imitation Learning), a method that extends the success of datamodels to robotics by addressing the unique challenges of data-driven data selection. DataMIL trains a data-quality estimator using a tractable surrogate objective. We empirically show that this objective retains sufficient correlations to the true objective (policy performance), allowing us to train datamodels that predict data influence without requiring expensive real-world rollouts. Moreover, thanks to a process that remains closer to the true objective, DataMIL selects and curates datasets for hard cases (e.g., different embodiments, multi-task settings) where prior heuristic-based methods degrade. Across 50 MetaWorld [23] tasks, we show a 10% performance boost compared to state-of-the-art baselines. We then show how datamodels can be estimated efficiently for large policies such as Octo [4] using improved datamodel estimators based on metagradients [24] and show its efficacy in curating task-specific datasets in ten tasks from the LIBERO benchmark [25] and four tasks in the real-world, leveraging OXE [3] even for new tasks and embodiments.

2 Related Work

Data curation for robot learning. Recent advances in robotics have leveraged ever-larger demonstration collections—both in simulation [26–28], and on real hardware [11–14, 29–34]—to train generalist policies capable of tackling diverse tasks [1–7]. However, the sheer scale and heterogeneity of these datasets (varying robots, scenes, and objectives) has motivated a body of work on data curation. For generalist training, methods like Re-Mix [35] use DoReMi [36] style optimization to learn optimal mixtures of data domains for improving model training, while others identify "high-quality" trajectories via mutual information criteria [37] or by scoring samples with policy rollouts [38]. Beyond generalist policy training, many studies have focused on task-specific dataset selection: given a handful of target demonstrations, one can sub-sample large datasets based on visual similarity [19], motion cues [39], or state–action closeness [21]. While these approaches capture human notions of quality, they remain agnostic to each sample's actual impact on downstream policy performance. Concurrent to our work, CUPID [40] focuses on single task curation by using a policy-gradient influence measure estimated via online rollouts. In contrast, DataMIL—built on the datamodels framework—demonstrates selection from large heterogeneous datasets, scoring datapoints entirely in an offline manner. This allows us to estimate each datapoint's contribution to final task success and curate training sets in an end-to-end, performance-aware fashion.

Datamodels and data attribution. Our work draws from a line of work in machine learning on data attribution [41–46] and, in particular, the datamodels framework [22, 44]. At a high level, this framework seeks to predict the behavior of machine learning models as a function of the data they are trained on. While we are not aware of work that has applied data attribution to robot learning, similar ideas have been explored for improving language model pre-training [47] and instruction tuning [48, 49, 24]; for increasing worst-group robustness [50]; and for removing outliers in supervised learning settings [51]. Our work builds on this body of research, while—as we discuss in Sec. 4.2—also tackling the unique challenges posed by the robot learning setting.

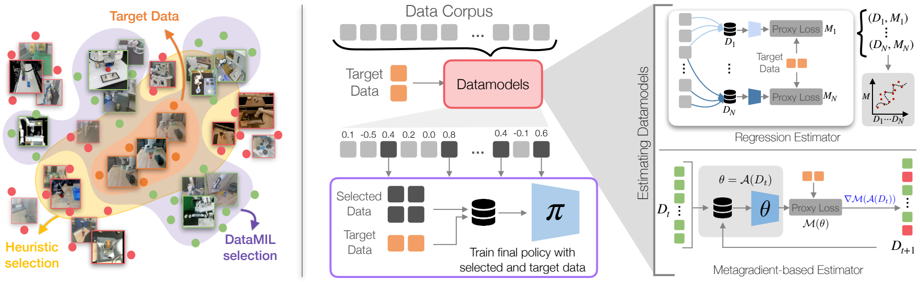


Figure 1: **Data selection with datamodels.** (left) Similarity-based methods select *close* samples (yellow), but these aren't always beneficial for learning. DataMIL evaluates data based on its impact on policy performance, selecting the samples that lead to policy improvement. (center) We estimate datamodels that score each sample by its influence on policy performance and select the highest-scoring samples for training. (right) DataMIL explores two datamodel estimation methods, adapted to robotics: *regression* and *metagradient-based* estimation (see Sec. 4.1).

3 Preliminaries: Data Selection for Policies and Datamodels Formalism

In this section, we first provide some general background on the problem of data selection for robot learning, and then describe the datamodels framework on which our method is based.

Policy learning. The focus of our work is on the *imitation learning* problem. Here, our goal is to learn a policy π that maps states s to distributions over actions a using a collection of training trajectories (or *demonstrations*) $\mathcal{D} = (\tau_1, \tau_2 \cdots \tau_n)$ of state-action pairs. We will define a policy learning *algorithm* $\mathcal{A}$ as a function that takes as input a dataset of demonstrations $\mathcal{D}$ and outputs an optimized policy π. We measure the performance of the policy π using a *metric* $\mathcal{M} : \pi \to \mathbb{R}$, which is a function mapping policies to a scalar value. The most common choice of metric is *success rate*—the fraction of times that sampling actions from the policy results in the policy completing a given task—but our notation is general and can also capture other choices of metric $\mathcal{M}$.

Data selection in robotics. In data selection for robotics, we are given (a) a prior dataset $\mathcal{D}$ of demonstrations; (b) a fixed learning algorithm $\mathcal{A}$ (e.g., stochastic gradient descent on an imitation learning objective); and (c) a target metric $\mathcal{M}$ that we will use to measure policy performance. Our goal is to select a subset of the data $\mathcal{D}' \subset \mathcal{D}$ such that applying the algorithm $\mathcal{A}$ to the subset $\mathcal{D}'$ yields a policy π that maximizes the target $\mathcal{M}$. Formally, we aim to find

$$\arg\max_{\mathcal{D}' \subset \mathcal{D}} \mathcal{M}(\mathcal{A}(\mathcal{D}')). \tag{1}$$

Solving this optimization problem is challenging since the algorithm $\mathcal{A}$ itself is usually expensive to compute (it involves, for example, training an imitation learning policy on $\mathcal{D}'$). Thus, exhaustive search over all subsets of $\mathcal{D}$ is infeasible.

Datamodels. In this work, we leverage data attribution [41, 42, 44, 52] (and specifically, the datamodels framework [22]) to tackle the data selection problem (1). The key idea behind datamodels is to directly approximate the target metric $\mathcal{M}(\mathcal{A}(\mathcal{D}'))$ as a function of the data $\mathcal{D}'$, allowing us to answer questions like "what would the performance of the policy be if we trained on this subset of the data?" without actually training the policy. More precisely, a datamodel is a model $\hat{f} : 2^{\mathcal{D}} \to \mathbb{R}$ that takes as input a subset of the data $\mathcal{D}' \subset \mathcal{D}$ and outputs an estimate of $\mathcal{M}(\mathcal{A}(\mathcal{D}'))$.

Informal Definition 1 (Datamodeling problem) *Given a prior dataset $\mathcal{D}$ and a learning algorithm $\mathcal{A}$, the* datamodeling problem *is the problem of predicting the behavior of a model trained on a subdataset $\mathcal{D}' \subset \mathcal{D}$ without actually training a model.*

If we had such an approximation in hand (assuming for now that we are able to compute one), we would approach the data selection problem (1) by solving the following optimization problem:

$$\arg\max_{\mathcal{D}' \subset \mathcal{D}} \hat{f}(\mathcal{D}'). \tag{2}$$

This problem is much more tractable than the original problem (1) because the datamodel $\hat{f}$ is typically much cheaper to compute than the learning algorithm and target metric $\mathcal{M}(\mathcal{A}(\cdot))$. For example, in supervised learning, recent works have shown that even *linear* datamodels—functions $\hat{f}$ that decompose additively in terms of their inputs $\mathcal{D}$—can be accurate predictors of model performance [22, 51, 44, 52, 45]. Intuitively, this means that we can assign a scalar value to each data point in $\mathcal{D}$ that indicates how much it contributes to the performance of the learning algorithm $\mathcal{A}$.

Since leveraging an accurate datamodel $\hat{f}$ for data selection is straightforward, the main challenge in our work is *constructing* such a datamodel. That is, we need to find a way to build a function $\hat{f}$ that can predict, with nontrivial accuracy, the performance of the learning algorithm $\mathcal{A}$ on any given subset of the data $\mathcal{D}'$ *without actually training a policy on that subset*. In the next section, we describe our methods for constructing such a datamodel in the policy learning setting.

4 DataMIL: Datamodels for Robot Imitation Learning

An overview of our training and selection methodology is shown in Figure 1. Below, we provide the details of each component, beginning with the estimation of datamodels.

4.1 Estimating Datamodels

In our work, we consider two ways of estimating datamodels: the *regression* method [22] and the *metagradient* method [24]. Both estimators approximate the outcome of model training *linearly*, in the sense that, for any training subset $\mathcal{D}' \subset \mathcal{D}$, the datamodel prediction $\hat{f}(\mathcal{D}')$ takes the form

$$\hat{f}(\mathcal{D}') = \sum_{z_i \in \mathcal{D}'} \tau(z_i).$$

Intuitively, $\tau(z_i)$ captures the importance of the training example z_i to the target metric $\mathcal{M}(\mathcal{A}(\mathcal{D}'))$ (more precisely, $\tau(z_i)$ is the *additive effect* of z_i on the target metric). Linear datamodels are convenient in the context of data selection, since they allow us to solve (1) by simply selecting the training examples with the highest scores $\tau(z_i)$. Both estimators below take this form—the only difference is in how they compute the $\tau(z_i)$ terms.

Regression estimator. The regression estimator is a straightforward but expensive way to estimate datamodels—it involves *precomputing* the scores $\tau(z_i)$ for each training example z_i in the broader training set $\mathcal{D}$. Concretely, we first sample N random subsets of the prior dataset $\mathcal{D}_j \subset \mathcal{D}$; for each of these datasets, we train a policy $\mathcal{A}(\mathcal{D}_j)$, and evaluate the target metric $\mathcal{M}$. (In our setting, evaluating $\mathcal{M}$ means rolling out the policy several times and computing the success rate.) We compute the scores $\tau(z_i)$ for all the training examples by solving the following minimization problem:

$$\{\tau(z_1), \ldots, \tau(z_n)\} := \arg\min_{\boldsymbol{\tau} \in \mathbb{R}^n} \sum_{j=1}^{N} \left(\sum_{j : z_i \in D_j} \tau_i - \mathcal{M}(\mathcal{A}(D_j)) \right)^2. \tag{3}$$

Above, observe that the sum $\sum_{j : z_j \in D_i} \tau_j$ is precisely the datamodel prediction of $\mathcal{M}(\mathcal{A}(D_i))$ when setting $\tau(z_j) = \tau_j$. Thus, (3) corresponds exactly to linearly regressing the target metric onto the presence of each training point z_i.

Metagradient-based estimator. While the approach above produces accurate datamodels $\hat{f}$ [22], it requires us to train thousands of policies on subsets of the prior dataset $\mathcal{D}$. This requirement makes it difficult to scale the regression-based estimator to finetuning large visuomotor policies such as Octo [4]. Fortunately, a line of work in computer vision and language modeling has devised far more efficient datamodel estimators that still accurately predict model behavior [44, 52, 46]. We adopt one such estimator for our purpose referred to as the metagradient-based estimator [24, 46], which enables efficient linear datamodel estimation when the target function $\mathcal{M}$ is differentiable with respect to the model parameters $\mathcal{A}(D)$. By devising a new way to compute a classical statistical quantity called the *influence function* [53, 41], metagradient-based estimator can almost *perfectly* predict the behavior of a model as a function of the training dataset at the cost of only a few model trainings on the prior dataset $\mathcal{D}$ [46]—we refer the reader to the Appendix or to [24] for more details.

4.2 DataMIL: Adapting Datamodels for Robotics

While datamodels have been applied to language modeling [44, 52] and computer vision [41, 22] tasks, there are some unique challenges that we face when applying them to robotics. Below, we describe how DataMIL extends the datamodel framework to handle robotic datasets efficiently.

Estimating datamodels without rollouts. In principle, the ideal target metric $\mathcal{M}$ is the policy's true success rate under environment rollouts. However, real-world rollouts are expensive, and using them directly renders the objective non-differentiable, preventing the use of estimators like Metagradients that exploit differentiability of the evaluation metric. To overcome these limitations, we introduce a proxy metric $\widehat{\mathcal{M}}$ that (1) requires no additional rollouts and (2) is fully differentiable. Concretely, given a small held-out demonstration set $\mathcal{D}_{target}$ for the target task, we define:

$$\widehat{\mathcal{M}}(\pi, \mathcal{D}_{target}) = \frac{1}{|\mathcal{D}_{target}|} \sum_{(s,a)\in\mathcal{D}_{target}} -\mathcal{L}_{BC}(\pi(s), a) \tag{4}$$

Where $L_{BC}(\pi(s), a)$ defines the policy loss on a training example (s, a) (see App. C for the exact objective for different policy classes). Hence, the true target metric $\mathcal{M}$ can be substituted by the proxy metric $\widehat{\mathcal{M}}$ in our original optimization (Eq.1), resulting in a more tractable and end-end differentiable objective for applying datamodels to robotic settings.

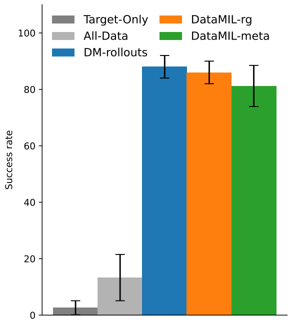


Figure 2: Comparing true rollout success ($\mathcal{M}$) vs. proxy metric ($\widehat{\mathcal{M}}$)

A natural question is whether $\widehat{\mathcal{M}}$ is a sufficiently good approximation of the true target to enable data selection. We study this question using the `pick-place-wall` task from MetaWorld [23], where we consider three different datamodel estimation techniques:

i) *DM-rollouts*: using regression-based estimator to estimate a datamodel for the "true" target $\mathcal{M}$ (success rate across rollouts);

ii) *DataMIL-rg*: using regression-based estimator to estimate a datamodel for proxy target $\widehat{\mathcal{M}}$ (loss on a heldout validation set);

iii) *DataMIL-meta*: using metagradient-based estimator to estimate a datamodel for proxy target.

We use each of these datamodels to select the top 10% of samples (as ranked by their estimated coefficient $\tau(z_i)$) from a prior dataset consisting of a mix of expert and suboptimal demonstrations (see Section 5 for details). We then measure the *true success rate* of a policy trained on the selected samples, and visualize the results in Fig. 2. Our results show that (a) selecting data for the proxy metric $\widehat{\mathcal{M}}$ incurs only a marginal drop in final success; (b) applying the metagradient-based estimator only incurs another small drop in success rate but are significantly faster to train. Moreover, all of the policies trained on selected data vastly outperform baselines trained on (i) the entire dataset or (ii) $\mathcal{D}_{target}$ alone—achieving up to 7× higher success than the all-data policy, while the target-only policy fails almost entirely.

These results demonstrate that (a) our proposed proxy objective can effectively stand in for expensive rollouts, and (b) data curation is critical: naively using all data or only target examples yields weak policies, whereas our curated datasets substantially boosts task success.

Clustering training examples. Datamodels estimate how each training sample—or training example—influences policy performance. In robotics, where data is naturally sequential, training examples may be individual state–action pairs or temporally extended sequences. Estimating influence at the single-sample level can be noisy, especially in large datasets where each sample is seen infrequently during training. To mitigate this, we group data into temporal clusters and measure influence at the cluster level. In practice, the optimal granularity depends on dataset size and compute budget: fine-grained clusters offer precise selection but suffer from higher estimation noise, while coarse clusters yield more stable influence estimates at the cost of detail.

Reducing distribution shift. Distribution shift—especially common in robotics due to changes in lighting, camera pose, or robot dynamics—poses a major challenge when selecting from large, diverse datasets. Since datamodels rely on training policies to estimate influence on target performance, a large mismatch between prior and target data can degrade the quality of datamodel estimation. To mitigate this, we include half of the $\mathcal{D}_{target}$ dataset alongside the prior data during datamodel estimation to better align the policy learning with the target domain, while using the other half purely for proxy objective evaluation. We only apply this in real-world (i.e., OXE), where the distribution gap is significant; in simulation, the prior and target data are typically well-aligned.

4.3 Data Selection and Policy Training

By applying the datamodel estimators described in Section 4.1 utilizing our proposed modifications in Section 4.2 we obtain a per-cluster influence score on policy performance. While these influence scores can be used in a variety of different ways, in this work we use them to curate a training subset: we select the top $x\%$ of prior examples with the highest positive influence to form $\mathcal{D}_{sel}$. We then train the downstream policy π via behavior cloning on $\mathcal{D}_{sel} \cup \mathcal{D}_{target}$ using a co-training recipe [20, 54, 13, 55]: at each training step, we sample from $\mathcal{D}_{target}$ with probability α and from $\mathcal{D}_{sel}$ with probability $1 - \alpha$. We found $\alpha = 0.5$ to perform well across all our experiments.

In summary, DataMIL leverages datamodels [22] to estimate how individual training examples affect a policy's performance on a given task. We propose key modifications that make this estimation tractable and robust in robotics settings—reducing noise, avoiding expensive rollouts, and scaling to large models and datasets. Given a prior dataset $\mathcal{D}$ and target dataset $\mathcal{D}_{target}$, we **(1) Cluster** the prior data $\mathcal{D}$ into trajectories or sub-trajectories, **(2)** Estimate influence scores using our proposed **proxy metric**, with the *regression* or *metagradient* datamodel estimators, **(3)** Select the top ranked clusters and create $\mathcal{D}_{sel}$, and **(4)** Train a final policy co-trained on the target and selected data.

5 Experiments

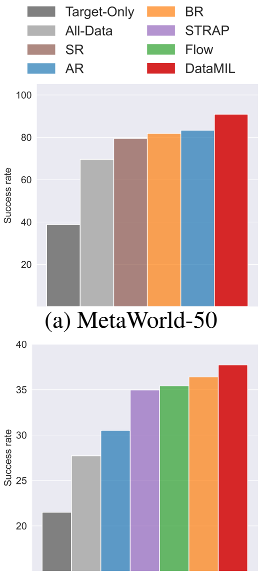
Datasets. We test DataMIL on two widely used multi-task simulation benchmarks: (1) MetaWorld [23], contains a suite of 50 distinct robot manipulation tasks, on a 7-DoF Sawyer robot arm, and (2) LIBERO benchmark [25], consists of 100 tasks with diverse objects, layouts and scenes. In the real world we test using the Open-X Embodiment (OXE) datasets [3] – an aggregation of many diverse robotic datasets collected across various robots and labs around the world.

Training and Evaluation Details. We use the language-conditioned Octo [4] model in LIBERO and OXE settings, initializing the model with a pretrained checkpoint provided by the authors to speed up training. For MetaWorld, we use the environment state as policy input, and hence use a simpler MLP based policy with a Gaussian action head from garage [56], and study both goal-conditioned and no-conditioning settings. Results for the latter can be found in the App. F.

Baselines. We compare DataMIL to prior works that select data using similarity based heuristics: BehaviorRetreival **(BR)** [21] trains a VAE on state-action pairs and uses similarity with the target data in the latent space to retrieve single state-action pairs; FlowRetrieval **(Flow)** [20] uses a similar approach but trains the VAE on the flow features of the images computed using GMFlow [39]; **STRAP** [19] uses features from a pretrained DinoV2 [57] model and uses dynamic time-warping over sub-trajectories to retrieve similar segments. We also introduce a simple action retrieval (**AR**) heuristic that computes similarity between action sequences of target and prior demonstrations and retrieves the most similar samples. Finally, we also train policies only on the target data (**Target-Only**), and co-trained with all data (**All-Data**) to measure the overall importance of data selection.

5.1 Results

How does data selected using DataMIL impact policy performance? **Metaworld.** MetaWorld's 50 manipulation tasks offer a rigorous testbed for data selection. We construct our prior dataset $\mathcal{D}$ by combining (1) expert demonstrations generated by scripted policies and (2) lower-quality exploration trajectories sampled from the replay buffer of a multi-task SAC agent trained across all tasks (see App. E.1 for details). For each task, we use 5 expert demos as $\mathcal{D}_{target}$, and use the regression-based datamodel estimator to select the top 10% of samples (Sec. 4)



(a) MetaWorld-50

(b) LIBERO-10

Figure 3: Avg. policy performance trained on selected datasets in sim. environments.

This setup is challenging as the selection method must both identify relevant tasks and filter noisy, suboptimal actions from the autonomous data. In Fig. 3a, we report policy performance averaged over all 50 tasks. Similarity-based baselines perform poorly: state-only retrieval (**SR**) fails to reject poor actions, action-only retrieval (**AR**) selects irrelevant tasks with similar action distributions, and state-action retrieval (**BR**) gives equal weight to both modalities, which may not be the appropriate recipe for all tasks. In contrast, by directly estimating each sample's influence on downstream policy performance, DataMIL robustly identifies useful demonstrations and discards harmful samples.

Can we scale DataMIL to larger and more complex policy classes?

LIBERO. We test DataMIL on 10 long-horizon tasks from the LIBERO-10 setting, using LIBERO-90 (comprising 4500 human-teleoperated demonstrations) as the prior dataset and selecting 10% of the data. The complex tasks and the high-dimensional RGB observations in LIBERO demand a powerful policy; we use Octo [4], a transformer-based diffusion policy. Training Octo is costly, making the "regression" datamodel estimator (which requires retraining across many subsets) impractical, and so we employ the metagradient datamodel estimator (Section 4.1).

Fig. 3b compares success rates of policies trained on the data selected by DataMIL with the baselines in each of the 10 target tasks. The relatively clean structure of LIBERO—single-view, consistent embodiment—makes it favorable for baselines that select via visual similarity (eg. **STRAP**, **BR** and **Flow**). However, we observe that their effectiveness varies significantly across tasks, likely due to the task-dependent suitability of each heuristic. In contrast, DataMIL consistently performs well across all tasks, achieving the highest average performance overall.

Can we select data from large heterogeneous datasets in the real world?

Open X-Embodiment Dataset (OXE). In the real world, we show data selection from the OXE [3] datasets and evaluate on four tasks on two robot embodiments shown in Figure 4 (top). This setting is particularly challenging: OXE is a heterogeneous aggregation of data from different labs, robots, camera setups, lighting conditions, and object arrangements. Further, none of our test tasks appear in OXE—we avoid matching the scene, camera pose, or objects. Instead, we aim to understand whether seemingly unrelated prior data can still yield positive transfer when curated appropriately. Our base setup uses 24 OXE datasets part of Octo's original training [4]. For **Franka-Ball** and **Franka-Pouch**, we subset this to 13 and 23 datasets, respectively (denoted as OXE-13 and OXE-23). Details on size of D_{target}, dataset partitions and evaluation methodology are provided in App. E.3.

Due to the scale of OXE, we replace the **All-Data** baseline with a **Random** baseline that samples the same number of datapoints as DataMIL and other methods. In Figure 4, we observe that DataMIL effectively selects relevant data even from highly heterogeneous sources. In the simpler **Franka-Ball** task, visual similarity-based baselines perform competitively. However, as the dataset grows more diverse—as in **Franka-Pouch**—these heuristics begin to break down, while DataMIL continues to identify data that improves policy performance. In the **Tiago-Sink** task, we explore a harder setting, selecting data for the Tiago [58] robot, an embodiment that never appears in the prior data. Despite this, DataMIL is able to select cross-embodiment demonstrations that improve task success. We discuss this further in Section 5.2. Finally, we move beyond single-task selection in the **Droid-Multitask** setting, where the target comprises three tasks: *bread in bowl*, *napkin in drawer*, and *open drawer*. This setting tests whether a single curated dataset can support multiple downstream

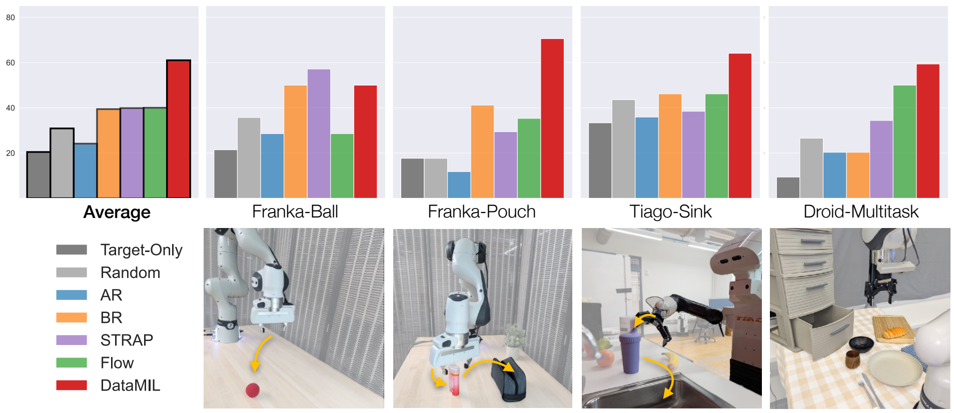


Figure 4: **Results for data selection on OXE.** We test the performance of policies trained on data selected from the Open X-Embodiment dataset using different selection strategies. Data selected using DataMIL achieves the highest performance across all tasks, highlighting the need for end-end policy-aware data selection techniques. (**Droid-Multitask** shows the average success rate across all its tasks. Individual task success rates shown in App. F)

objectives simultaneously. DataMIL consistently outperforms baselines, selecting examples that improve its performance on all target tasks and yielding stronger average overall.

These results highlight that DataMIL scales to real-world robotics, handles heterogeneous datasets, and supports both single-task and multitask learning—even in settings with unseen embodiments.

5.2 What data is selected by DaMIL?

We briefly discuss the DataMIL-selected data; see appendix for visualizations and further insights.

Type of embodiments selected. DataMIL is able to select useful data for a completely new embodiment in the **Tiago-Sink** experiment. In the appendix we show some snippets of the highest frequency datasets selected by DataMIL. We observe that even though they are visually quite different, sampled from datasets such as RT-1 [1], BC-Z [30] and Bridge [11, 12], they still represent the essence of the target tasks – robots operating on a table top from an ego-perspective.

Distribution of selected data. We also notice that data selected by DataMIL usually spans several different datasets, whereas most baselines select a majority of their data from a single dataset (plots shown in appendix). We hypothesize that since there is no data that exactly matches the target task, the selected data must not only be relevant but general, so as to enable positive transfer in capabilities and not make the policy overfit to a single type of domain.

Top and bottom influencers. Analyzing the highest and lowest ranked datapoints by DataMIL, we find that they typically look similar (e.g., same embodiment or dataset). This actually makes sense: similar states can have very different action distributions, and while some of these actions might help reduce the policy loss on the target data, the others might lead to a large deviation, making them *harmful* for final policy learning. This aligns with data attribution works in computer vision, where harmful data looks very similar to helpful data but with a different label [22, 59].

6 Conclusion

We present DataMIL, a data-driven method for data selection for imitation learning. DataMIL builds upon the framework of datamodels, which has been applied successfully to data selection in NLP and CV, and extends it to real-world robotic applications. Extensive experiments in simulation and real-world settings empirically support that DataMIL retrieves data to train higher-performing policies than multiple existing state-of-the-art baselines, particularly in complex scenarios. Overall, we see DataMIL as a step towards better-aligned data curation methods for robotics.

Acknowledgments

We thank Luca Macesanu for helping with some of the real world evaluations and Bowen Jiang for his feedback on the manuscript. We also thank Julian Yapeter and Karl Pertsch for their assistance in generating the MetaWorld dataset. Work supported in part by DARPA TIAMAT program (HR0011-24-9-0428) and also supported by the NSF grant DMS-2134108 and Open Philanthropy.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[2] 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.

[3] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[4] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[6] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[7] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[8] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[9] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[10] J. Huang, S. Yong, X. Ma, X. Linghu, P. Li, Y. Wang, Q. Li, S.-C. Zhu, B. Jia, and S. Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023.

[11] 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. *arXiv preprint arXiv:2109.13396*, 2021.

[12] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[13] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[14] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.

[15] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[16] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[17] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[18] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. G. Arenas, A. Zeng, F. Xia, and D. Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15172–15179. IEEE, 2024.

[19] M. Memmel, J. Berg, B. Chen, A. Gupta, and J. Francis. Strap: Robot sub-trajectory retrieval for augmented policy learning. *arXiv preprint arXiv:2412.15182*, 2024.

[20] L.-H. Lin, Y. Cui, A. Xie, T. Hua, and D. Sadigh. Flowretrieval: Flow-guided data retrieval for few-shot imitation learning. *arXiv preprint arXiv:2408.16944*, 2024.

[21] M. Du, S. Nair, D. Sadigh, and C. Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *arXiv preprint arXiv:2304.08742*, 2023.

[22] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Predicting predictions from training data. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[23] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[24] L. Engstrom, A. Ilyas, B. Chen, A. Feldmann, W. Moses, and A. Madry. Optimizing ml training with metagradient descent. *arXiv preprint arXiv:2503.13751*, 2025.

[25] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[26] 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. *arXiv preprint arXiv:2108.03298*, 2021.

[27] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[28] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[29] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[30] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[31] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[32] P. Sharma, L. Mohan, L. Pinto, and A. Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[33] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[34] S. Dass, K. Pertsch, H. Zhang, Y. Lee, J. J. Lim, and S. Nikolaidis. Pato: Policy assisted teleoperation for scalable robot data collection. *arXiv preprint arXiv:2212.04708*, 2022.

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

[36] S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. S. Liang, Q. V. Le, T. Ma, and A. W. Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

[37] 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. *arXiv preprint arXiv:2502.08623*, 2025.

[38] A. S. Chen, A. M. Lessing, Y. Liu, and C. Finn. Curating demonstrations using online experience. *arXiv preprint arXiv:2503.03707*, 2025.

[39] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8121–8130, 2022.

[40] C. Agia, R. Sinha, J. Yang, R. Antonova, M. Pavone, H. Nishimura, M. Itkina, and J. Bohg. Cupid: Curating data your robot loves with influence functions. *arXiv preprint arXiv:2506.19121*, 2025.

[41] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[42] A. Ilyas, K. Georgiev, L. Engstrom, and S. M. Park. Data attribution at scale. Tutorial at ICML 2024, 2024. URL https://ml-data-tutorial.org.

[43] Z. Hammoudeh and D. Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.

[44] S. M. Park, K. Georgiev, A. Ilyas, G. Leclerc, and A. Madry. TRAK: Attributing model behavior at scale. In *Arxiv preprint arXiv:2303.14186*, 2023.

[45] T. A. Chang, D. Rajagopal, T. Bolukbasi, L. Dixon, and I. Tenney. Scalable influence and fact tracing for large language model pretraining. In *The Thirteenth International Conference on Learning Representations*, 2024.

[46] A. Ilyas and L. Engstrom. Magic: Near-optimal data attribution for deep learning, 2025.

[47] L. Engstrom, A. Feldmann, and A. Madry. Dsdm: Model-aware dataset selection with datamodels. In *International Conference on Machine Learning*, pages 12491–12526. PMLR, 2024.

[48] M. Xia, S. Malladi, S. Gururangan, S. Arora, and D. Chen. Less: selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 54104–54132, 2024.

[49] Z. Liu, A. Karbasi, and T. Rekatsinas. Tsds: Data selection for task-specific model finetuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[50] S. Jain, K. Hamidieh, K. Georgiev, A. Ilyas, M. Ghassemi, and A. Madry. Improving subgroup robustness via data selection. *Advances in Neural Information Processing Systems*, 37:94490–94511, 2024.

[51] J. Lin, A. Zhang, M. Lécuyer, J. Li, A. Panda, and S. Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. In *International Conference on Machine Learning*, pages 13468–13504. PMLR, 2022.

[52] J. Bae, W. Lin, J. Lorraine, and R. Grosse. Training data attribution via approximate unrolled differentiation. *CoRR*, 2024.

[53] F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

[54] A. Maddukuri, Z. Jiang, L. Y. Chen, S. Nasiriany, Y. Xie, Y. Fang, W. Huang, Z. Wang, Z. Xu, N. Chernyadev, et al. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. *arXiv preprint arXiv:2503.24361*, 2025.

[55] S. Nasiriany, T. Gao, A. Mandlekar, and Y. Zhu. Learning and retrieval from prior data for skill-based imitation learning. *arXiv preprint arXiv:2210.11435*, 2022.

[56] T. garage contributors. Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage, 2019.

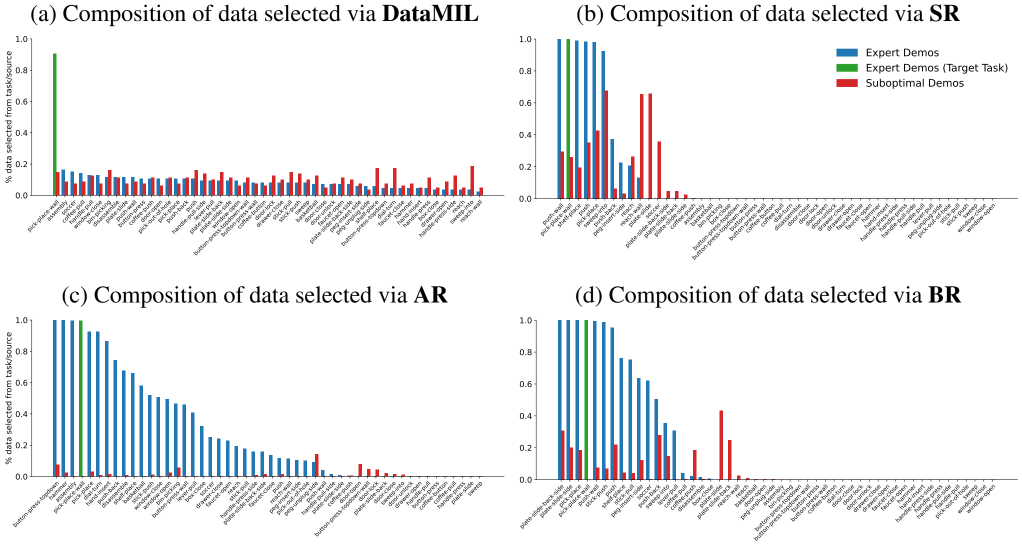
[57] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[58] J. Pages, L. Marchionni, and F. Ferro. Tiago: the modular robot that adapts to different research needs. In *International workshop on robot modularity, IROS*, volume 290, 2016.

[59] V. Feldman and C. Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

[60] S. Dass, W. Ai, Y. Jiang, S. Singh, J. Hu, R. Zhang, P. Stone, B. Abbatematteo, and R. Martín-Martín. Telemoma: A modular and versatile teleoperation system for mobile manipulation. *arXiv preprint arXiv:2403.07869*, 2024.

A Qualitative Results: What data is selected by DataMIL?



(a) Composition of data selected via **DataMIL**

(b) Composition of data selected via **SR**

(c) Composition of data selected via **AR**

(d) Composition of data selected via **BR**

Figure 5: **MetaWorld Qualitative Results.** Percentage of data selected from each task and expert/suboptimal source for **DataMIL**, **SR**, **AR** and **BR**

A.1 MetaWorld

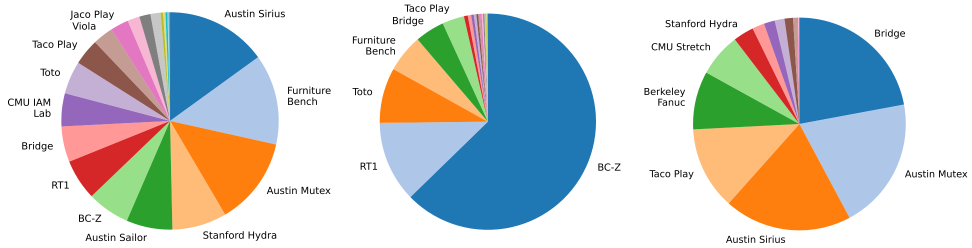
As described in Appendix E.1, our prior dataset for MetaWorld combines both expert and sub-optimal demonstrations. In our experiments, we retrieve the top 10% of this data ranked by DataMIL to train the policy. Here, we qualitatively examine the dataset selected by DataMIL for the **pick-place-wall** task.

Figure 5 shows the percentage of data selected from each task and data source (expert or suboptimal). We observe that **SR**, while able to retrieve samples from relevant tasks, fails to differentiate between expert and sub-optimal demonstrations—resulting in the inclusion of a large fraction of low-quality data. In contrast, **AR** filters out sub-optimal samples more effectively by matching actions, but it lacks task awareness due to its disregard for state information, often pulling data from irrelevant tasks. **BR**, which embeds both state and action features jointly, exhibits a blend of SR and AR behaviors—capturing elements of both but also inheriting their limitations. In comparison, **DataMIL** consistently selects data from the correct task (green bar) while also avoiding noisy, sub-optimal examples.

A.2 OXE

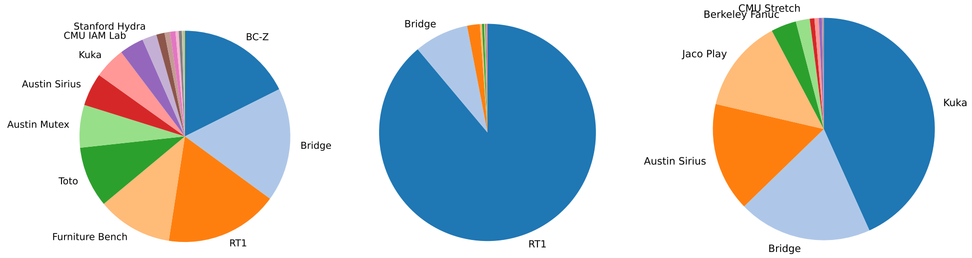
Here we elaborate on our discussion of the qualitative analysis of the data selected by DataMIL in Section 5.2.

Distribution of Selected Data. In Figures 7 and 6, we show the distribution of datasets selected by DataMIL and representative baselines for the **Tiago-Sink** and **Franka-Pouch** tasks, respectively. As discussed in the main text, baseline methods often focus heavily on a single dataset. For example, **AR** retrieves most of its data from *RT-1* in the **Tiago-Sink** task (Figure 7b), while **Flow** disproportionately selects samples from *BC-Z* for **Franka-Pouch** (Figure 6b). In contrast, DataMIL consistently selects data across a broader range of datasets (Figure 7a and 6a). We hypothesize that this diversity is beneficial when no single prior dataset closely matches the target task—encouraging generalization rather than overfitting to a particular domain.



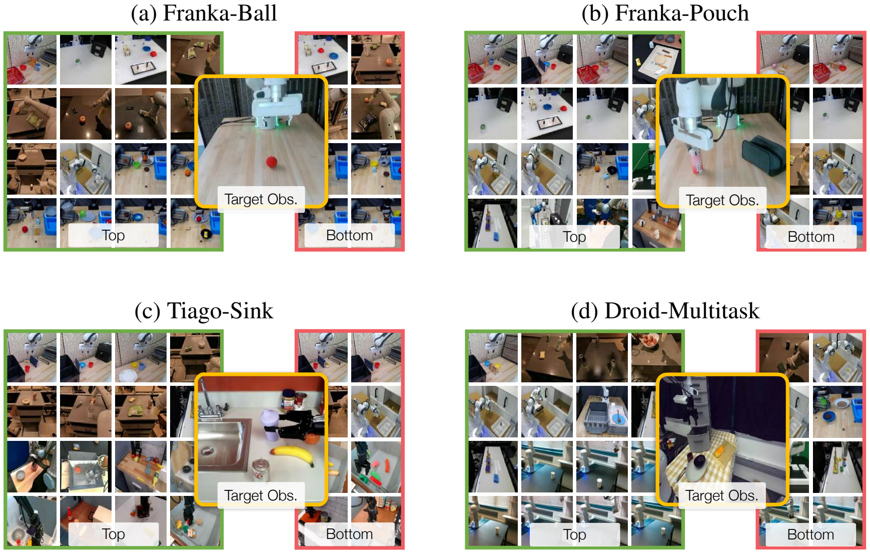
(a) Data selected by **DataMIL** (b) Data selected by **Flow** (c) Data selected by **BR**

Figure 6: Distribution of datasets selected by different methods for **Franka-Pouch** task



(a) Data selected by **DataMIL** (b) Data selected by **AR** (c) Data selected by **BR**

Figure 7: Distribution of datasets selected by different methods for **Tiago-Sink** task



(a) Franka-Ball (b) Franka-Pouch (c) Tiago-Sink (d) Droid-Multitask

Figure 8: Top and bottom ranked samples by **DataMIL** for each of the real-world tasks

Type of embodiments selected. In the **Tiago-Sink** task (Figure 7a), DataMIL successfully selects data for a novel embodiment (Tiago), which does not appear in prior data. The selected samples span datasets such as *RT-1*, *BC-Z*, and *Bridge*, all of which involve tabletop manipulation from an egocentric viewpoint—capturing task-relevant structure even without embodiment match. For baselines, we observe that even when the target embodiment is present in the prior dataset (e.g., Franka), the selected data often comes from unrelated domains. For instance, in **Franka-Pouch**

(Fig. 6), **Flow** selects data from *BC-Z* (Google Robot), and **BR** retrieves from *Bridge* (WidowX). In contrast, the top five most frequently selected datasets by DataMIL are all sourced from Franka.

Top and bottom influences. Figure 8 visualizes the highest and lowest ranked datapoints identified by DataMIL for real-world tasks. Interestingly, these examples often appear visually similar, aligning with findings in computer vision [22, 59] where data points that look alike but carry different labels can mislead the model. In robotics, this may occur when visually similar observations correspond to different actions, thereby confusing the policy. This could also be an artifact caused by the sensitivity of validation loss used in DataMIL's proxy metric and understanding how and why these fine-grained differences affect data selection, and ultimately, the policy performance, is an interesting direction for future work.

B Estimating Datamodels

In this section, we describe the datamodeling framework in more detail. In particular, we first provide the formal version of Informal Definition 1, then describe the estimators that we use to construct datamodels in this work, namely the regression estimator and the metagradient-based estimator.

B.1 Formalizing Datamodeling

The goal of datamodeling is to construct a function $\hat{f}$ that can predict the performance of a learning algorithm $\mathcal{A}$ on any given subset of the data $\mathcal{D}' \subset \mathcal{D}$ without actually training a policy on that subset. Let $\mathcal{D}$ be a prior dataset of imitation-learning data, and let us represent any subset of $\mathcal{D}$ as a binary vector $\mathbf{z} \in \{0,1\}^N$ where $z_i = 1$ if the i-th training sample is in the subset and $z_i = 0$ otherwise. Let a learning algorithm $\mathcal{A}$ be a function that takes as input a dataset (represented as a binary vector $\mathbf{z}$) and outputs a policy $\mathcal{A}(\mathbf{z})$. The datamodeling problem is to construct a function $\hat{f}$ that can predict the performance of $\mathcal{A}(\mathbf{z})$ when trained on any given subset of the data without actually training a policy on that subset. More formally, we aim to find a function $\hat{f}$ minimizing the following loss:

$$\mathbb{E}_{\mathbf{z}\sim \text{Bernoulli}(\frac{1}{2})^N} \left[\left(\mathcal{M}(\mathcal{A}(\mathbf{z})) - \hat{f}(\mathbf{z}) \right)^2 \right], \tag{5}$$

where $\mathcal{M}$ is the target metric and $\mathbf{z} \sim \text{Bernoulli}(\frac{1}{2})^N$ is a random binary vector of length N.

Recall from the main text that we are particularly interested in datamodels $\hat{f}$ that are additive in the training dataset—in terms of our formalization, we are interested in functions $\hat{f}$ of the form

$$\hat{f}(\mathbf{z}) = \mathbf{z}^\top \beta. \tag{6}$$

for some vector $\beta \in \mathbb{R}^N$. An *estimation method* for datamodels is thus just a method for finding a good estimate of the vector β. We refer the reader to Ilyas et al. [22] for a more detailed discussion of datamodeling.

B.2 Regression Estimator

The regression estimator is a simple yet effective method for estimating the vector β that treats datamodeling as a supervised learning problem. In particular, the regression estimator first samples a set of m binary vectors $\mathbf{z}_1, \ldots, \mathbf{z}_m \sim \text{Bernoulli}(\frac{1}{2})^N$; for each of these binary vectors, it trains a policy $\mathcal{A}(\mathbf{z}_i)$ on the subset of the data indexed by $\mathbf{z}_i$, and evaluates its performance using the target metric $\mathcal{M}$. It then fits a linear model to the performance of these policies on the sampled binary vectors, i.e., it solves

$$\min_{\beta\in\mathbb{R}^N} \frac{1}{m} \sum_{i=1}^{m} \left(\mathcal{M}(\mathcal{A}(\mathbf{z}_i)) - \mathbf{z}_i^\top \beta \right)^2, \tag{7}$$

and uses the resulting vector $\hat{\beta}$ as the parameters of the datamodel $\hat{f}(\mathbf{z}) = \mathbf{z}^\top \hat{\beta}$. The cost of building this estimator is high, since it requires training a policy for each of the m binary vectors, but Ilyas et al. [22] shows that the estimator can be very accurate, and indeed identifies highly influential subsets of the prior dataset.

B.3 Metagradient-based Estimator

The metagradient-based estimator is a more sophisticated method for estimating the vector β that avoids the high cost of training a policy for each binary vector. Instead, the metagradient-based estimator operates by leveraging a classical statistical tool called the *influence function* [53]. Intuitively, the metagradient-based estimator proceeds as follows:

1. First, instead of thinking of training datasets as *binary* vectors $\mathbf{z} \in \{0,1\}^N$, we think of them as *real-valued* vectors $\mathbf{z} \in [0,1]^N$, where each coordinate z_i corresponds to the *importance weight* placed on the i-th training sample in the dataset. Concretely, if $z_i = 0$, then the i-th training sample is not used in the training set, and if $z_i = 1$, then the i-th training sample is used in the training set with full weight; if $0 < z_i < 1$, then the i-th training sample is used in the training set but its loss is scaled by z_i. Observe that this parameterization is equivalent to the binary parameterization for $z_i \in \{0,1\}$, but gives us a continuous way to represent the training set.
2. Once we have this continuous parameterization, we can write the *first-order approximation* to $\mathcal{M}(\mathcal{A}(\mathbf{z}))$ as
$$\mathcal{M}(\mathcal{A}(\mathbf{z})) \approx \mathcal{M}(\mathcal{A}(\mathbf{z}_0)) + \nabla\mathcal{M}(\mathcal{A}(\mathbf{z}_0))^\top(\mathbf{z}-\mathbf{z}_0), \tag{8}$$
where $\mathbf{z}_0$ is the vector of all ones. The gradient $\nabla\mathcal{M}(\mathcal{A}(\mathbf{z}_0))$ is known as the *influence function*, and gives us a *linear* approximation to the loss in Equation 5. That is, if we could compute the influence function exactly, we could use it as a datamodel directly, i.e.,
$$\hat{f}(\mathbf{z}) = \mathcal{M}(\mathcal{A}(\mathbf{z}_0)) + \nabla\mathcal{M}(\mathcal{A}(\mathbf{z}_0))^\top(\mathbf{z}-\mathbf{z}_0). \tag{9}$$
3. Traditionally, the influence function is notoriously hard to compute, and so prior work on data attribution has focused on approximating it [41, 44, 52]. However, recent work has shown how to compute it *exactly* and *efficiently* [24] and how to use this exact influence function as a datamodel estimator [46].

Observe that in order for the metagradient-based estimator to be valid, the function $\mathcal{M}(\mathcal{A}(\mathbf{z}))$ must be *differentiable* with respect to $\mathbf{z}$. For this to be satisfied, it is sufficient for (a) the target metric $\mathcal{M}$ to be differentiable with respect to the policy $\mathcal{A}(\mathbf{z})$, and (b) the policy $\mathcal{A}(\mathbf{z})$ to be trained via an iterative algorithm composed of elementary differentiable operations (which is almost all of the popular off-the-shelf learning algorithms).

C Proxy Metric Details

Recall from Eq. 4 that our general proxy metric is

$$\widehat{\mathcal{M}}(\pi, \mathcal{D}_{target}) = \frac{1}{|\mathcal{D}_{target}|} \sum_{(s,a)\in\mathcal{D}_{target}} -\mathcal{L}_{BC}(\pi(s), a)$$

where $\mathcal{L}_{BC}$ is the behavior-cloning loss appropriate to the policy class. Here we provide how the equation looks like for specific policy classes that we used in our experiments.

MetaWorld. We parametarize the policy in MetaWorld as,

$$\pi_\theta(a \mid s) = \mathcal{N}\big(a;\, \mu_{\theta_1}(s),\, \mathrm{diag}(\sigma_{\theta_2}(s)^2)\big)$$

and consider two forms of $\mathcal{L}_{BC}$:

1. *Negative Log-Likelihood (NLL)*:
$$\begin{aligned}\mathcal{L}_{\text{NLL}}(s,a) &= -\log \pi_\theta(a \mid s) \\ &= \tfrac{1}{2}\,(a-\mu_{\theta_1}(s))^\top \Sigma(s)^{-1}\,(a-\mu_{\theta_1}(s)) \;+\; \tfrac{1}{2}\,\log\det\big(2\pi\,\Sigma(s)\big)\end{aligned} \tag{10}$$

2. l_1 *loss*:

$$\mathcal{L}_{\ell_1}(s,a) \;=\; \left\|a \;-\; \mu_{\theta_1}(s)\right\|_1 \tag{11}$$

Empirically, the $l1$ loss works better for the *regression estimator*, while *metagradient-based estimator* performs well with the NLL loss.

LIBERO and OXE. In the LIBERO and OXE settings, we use Octo as the learning model which is a transformer-based policy with a diffusion action head. It's behavior-cloning loss is the standard denoising score-matching objective:

$$\mathcal{L}_{\text{diff}}(s,a) = \mathbb{E}_{t\sim\text{Uniform}[1,T],\,\epsilon\sim\mathcal{N}(0,I)}\left\|\epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\,a + \sqrt{1-\bar{\alpha}_t}\,\epsilon,\; s,\; t\big)\right\|^2$$

where $\alpha_t \in (0,1)$ is the forward-process noise schedule, $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$, and $\epsilon_\theta(a_t,s,t)$ is the network's noise prediction.

Substituting $\mathcal{L}_{BC} = \mathcal{L}_{\text{diff}}$ into the proxy metric gives,

$$\widehat{\mathcal{M}}(\pi,\mathcal{D}_{\text{target}}) \;=\; -\,\frac{1}{|\mathcal{D}_{\text{target}}|}\sum_{(s,a)\in\mathcal{D}_{\text{target}}}\mathbb{E}_{t,\epsilon}\left\|\epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\,a + \sqrt{1-\bar{\alpha}_t}\,\epsilon,\; s,\; t\big)\right\|^2 \tag{12}$$

C.1 Up-weighting Relevant States

In robotics, more often than not we have some information about what states and actions are of higher importance than others, for example states closer to object interactions may be more relevant than moving around in free space. Our proxy metric provides a seamless way to incorporate prior knowledge by re-weighting important states in the behavioral cloning loss. Concretely, we introduce a state-action-dependent weight $w(s,a)$ into the objective:

$$\widehat{\mathcal{M}}(\pi,\mathcal{D}_{target}) = \frac{1}{|\mathcal{D}_{target}|}\sum_{(s,a)\in\mathcal{D}_{target}} -\underline{w(s,a)}\mathcal{L}_{BC}(\pi(s),a)$$

By default, we set $w(s,a) = 1$ and use the unweighted proxy. In the LIBERO experiments, however, we found that doubling the weight for states immediately preceding a grasp significantly improves data selection. Thus, for those "pre-grasp" states we use $w(s,a) = 2$, while all other states retain $w(s,a) = 1$.

D Baseline Implementation

Implementation details of the baselines are provided below.

- **FlowRetrieval [20] and BehaviorRetrieval [21]**: FlowRetrieval (**Flow**) and Behavior Retrieval (**BR**) baselines compute similarity on the image flows and state-action pairs respectively. Since these features typically include high-dimensional image observations, **Flow** and **BR** train VAEs to encode the features into a more manageable latent space, which they can use to compute similarities between prior and target data. We used the implementation provided by the authors of FlowRetrieval [20] for training the VAEs and computing the similarity for both, **Flow** and **BR**, in the LIBERO and OXE settings (`https://github.com/lihenglin/bridge_training_code`).

  For both **Flow** and **BR** (and other heuristics such as Action Retrieval (**AR**) and State Retrieval (**SR**)), once each state in the prior data is assigned a score based on the similarity measure, we select the top $x\%$ of the data most similar to the target where x is the selection budget.

- **STRAP [19]**: In the LIBERO experiments, we use the authors' **STRAP** implementation (`https://github.com/WEIRDLabUW/STRAP`) to embed sub-trajectories with DinoV2 [57] and compute similarity via dynamic time warping. **STRAP** expects HDF5-formatted inputs, but our OXE pipeline relies on TFDS. We therefore adapted the **STRAP** code to accept TFDS datasets without altering its core logic.

> While **STRAP**'s original recommendation is to retrieve the top 100 sub-trajectories in LIBERO, we found that training Octo on these segments underperforms. Instead, we retrieve the most similar sub-trajectories until they constitute 10% of the prior data—matching the budget used by DataMIL and our other baselines. In LIBERO, this modification boosts success from **24.72%** (with 100 segments) to **34.96%** averaged over LIBERO-10 tasks. We apply the same retrieval strategy on OXE, sampling sub-trajectories until we match the selection size of our method and baselines.

For MetaWorld, we found it more effective to compute similarity over temporal windows rather than individual state–action pairs. Specifically, for each baseline (**BR**, **SR**, **AR**), we slide a fixed-length horizon $\mathcal{H}$ over both prior and target data, concatenate each segment's states (and actions) into a single high-dimensional vector, and then measure similarity between these flattened vectors. This horizon-based approach captures temporal context, enabling the baselines to reject noisy or suboptimal samples—ultimately improving retrieval quality and downstream policy performance. We used $\mathcal{H} = 50$ for all tasks since our initial experiments found it to perform best.

E Training and Evaluation Details

E.1 MetaWorld

Dataset. The MetaWorld dataset is constructed from two sources: scripted expert policies and reinforcement learning (RL) exploration. MetaWorld provides scripted policies for each of its 50 tasks, which we use to generate 4,000 episodes totaling 350K environment steps. For the RL data, we train a multi-task SAC agent on all 50 tasks for 12 million transitions, reaching an average success rate of 21%. To create a representative prior dataset, we uniformly subsample from the SAC replay buffer across all tasks, yielding 1 million environment steps—approximately 3× larger than the scripted data. For each target task, we generate 5 expert demonstrations using the scripted policy as $\mathcal{D}_{target}$.

Datamodel estimation using DataMIL. We cluster the prior dataset at the trajectory level and use 5 demonstrations from $\mathcal{D}_{target}$ to compute the proxy objective. We then compute the datamodels using the *regression-based* datamodel estimator. The top 10% of prior trajectories, ranked by the datamodels, are selected to form $\mathcal{D}_{sel}$.

Policy Training and Evaluation. We train a behavior cloning policy with an MLP backbone and a tanh-squashed Gaussian output distribution on $\mathcal{D}_{sel}$. In preliminary experiments, we found that co-training with $\mathcal{D}_{target}$ yielded negligible improvements, so we exclude it in this setting. Each policy is trained and evaluated over 3 random seeds.

E.2 LIBERO

Dataset. Our prior dataset consists of 4,500 human teleoperated demonstrations from LIBERO-90, with 50 demonstrations per task. The 10 tasks from LIBERO-10 serve as our target tasks. For each, we randomly sample 5 demonstrations to form the target dataset $\mathcal{D}_{target}$.

Datamodel estimation using DataMIL. Prior to running DataMIL, we segment the prior demonstrations into sub-trajectories of horizon length 15, which we found to provide a good balance between granularity and noise-robustness. We then estimate influence scores using the *metagradient-based estimator* with the weighted proxy metric described in Appendix C.1. The top 10% of sub-trajectories, based on datamodel influence, are selected to form the selected dataset $\mathcal{D}_{sel}$.

Policy Training and Evaluation. We fine-tune a language-conditioned Octo policy, starting from the publicly released Octo-small checkpoint, by co-training on $\mathcal{D}_{target}$ and $\mathcal{D}_{sel}$ using a co-training ratio $\alpha = 0.5$ for 10k steps. We evaluate each policy on the corresponding target task using 50 rollouts and report results averaged over 5 random seeds.

Table 1: Datasets utilized across the OXE subsets in our experimental setup

Dataset	OXE13	OXE23	OXE24	Dataset	OXE13	OXE23	OXE24
RT1	✓	✓	✓	CMU IAM Lab	✓	✓	✓
Viola	✓	✓	✓	Roboturk	✗	✓	✓
Austin Buds	✓	✓	✓	BC-Z	✗	✓	✓
Austin Mutex	✓	✓	✓	CMU Stretch	✗	✓	✓
Austin Sailor	✓	✓	✓	DLR Edan	✗	✓	✓
Austin Sirius	✓	✓	✓	Berkeley Autolab UR5	✗	✓	✓
Taco Play	✓	✓	✓	Berkeley Fanuc	✗	✓	✓
Jaco Play	✓	✓	✓	Berkeley Cable	✗	✓	✓
Stanford Hydra	✓	✓	✓	Bridge	✗	✓	✓
NYU Franka	✓	✓	✓	NYU Door Opening	✗	✓	✓
Furniture Bench	✓	✓	✓	Toto	✗	✓	✓
UCSD Kitchen	✓	✓	✓	Kuka	✗	✗	✓

E.3 OXE

Table 2: Task-wise experimental setup for selecting data and, training and evaluating policies

	Embodiment	Prior	Prior Selection	No. of	No. of
		Dataset	Ratio	Target Demos	Evaluations
Franka-Pick	Franka-Panda	OXE13	1%	10	14
Franka-Pouch	Franka-Panda	OXE23	0.75%	30	17
Tiago-Sink	Tiago	OXE24	0.5%	20	39
Droid-Multitask	Franka-Panda	OXE24	1%	40 (total)	32 (total)
Drawer	-	-	-	10	10
Bread	-	-	-	15	12
Napkin	-	-	-	15	10

Dataset. We use subsets of the Open X-Embodiment (OXE) dataset [3] as our prior data. Specifically, we define three subsets—**OXE13**, **OXE23**, and **OXE24**—with their respective constituent datasets listed in Table 1. The mapping between tasks and dataset subsets is shown in Table 2. For each task, we collect a separate target dataset via teleoperation [60], varying the number of demonstrations per task based on difficulty (see Table 2).

Datamodel estimation using DataMIL. Following the DataMIL recipe, we cluster the prior data at the trajectory level and estimate influence using the metagradient-based estimator with our proposed proxy objective. To reduce distribution shift during datamodel training, we split $\mathcal{D}_{target}$ into two halves: one half is used to compute the proxy metric, while the other is included in the training mix. After training, we select the top $x\%$ of prior trajectories based on influence scores, where x is specified per task in Table 2.

Policy Training and Evaluation. We fine-tune a language-conditioned Octo-small checkpoint using co-training on $\mathcal{D}_{target}$ and the selected dataset $\mathcal{D}_{sel}$, with a co-training ratio $\alpha = 0.5$ for 50k steps. Final policy is evaluated on the corresponding target tasks using a fixed number of real-world rollouts (Table 2). To ensure fair comparisons, we fix the spatial configurations of relevant objects across all methods. For instance, in the **Franka-Pouch** task, we use 17 predefined object poses (position and orientation) for evaluation. In the more challenging **Droid-Multitask** setting, we report both full and partial successes as part of the final success rate (e.g., a partially closed drawer or grasping the bread/napkin), with partial completions weighted as 0.5. All real-world evaluations were conducted using a single random seed.

F Additional Results

Tasks	**Target-Only**	**All-Data**	**AR**	**BR**	**Flow**	**STRAP**	**DataMIL** (ours)
			LIBERO Evaluations				
Soup-Sauce	13.2 ± 7.6	20.0 ± 15.6	32.8 ± 9.3	38 ± 15.3	50 ± 14.4	33.2 ± 15.10.7	39.2 ± 11.9
Cream-Butter	27.6 ± 6.5	35.2 ± 13.2	39.6 ± 10.7	41.2 ± 14.6	47.2 ± 10.8	20.0 ± 9.7	50.4 ± 8.6
Stove-Moka	27.4 ± 7.3	24.0 ± 9.8	31.2 ± 8.1	30.4 ± 10.0	33.2 ± 6.7	43.6 ± 5.5	40.4 ± 8.5
Bowl-Cabinet	48.8 ± 6.4	65.6 ± 8.3	62.8 ± 3.3	73.6 ± 8.5	69.2 ± 12.2	77.2 ± 10.3	72.4 ± 5.2
Mug-Mug	0.4 ± 0.9	2.0 ± 2.44	4.8 ± 2.3	2.0 ± 2.8	6.4 ± 6.5	5.2 ± 5.2	0.8 ± 1.1
Book-Caddy	51.2 ± 8.3	58.4 ± 10.7	65.2 ± 13.1	76.8 ± 7.6	69.2 ± 13.2	83.2 ± 7.8	82.4 ± 1.7
Mug-Pudding	2.0 ± 2.0	2.4 ± 1.7	5.6 ± 5.9	8.4 ± 2.2	5.8 ± 3.5	10.8 ± 6.9	4.8 ± 3.3
Soup-Cheese	11.2 ± 3.3	22.0 ± 6.5	29.3 ± 10.4	35.2 ± 5.2	26.8 ± 4.8	35.2 ± 3.9	36.0 ± 6.0
Moka-Moka	5.6 ± 4.3	12 ± 8.2	4.8 ± 2.3	15.2 ± 4.1	7.6 ± 5.5	6.8 ± 1.8	12.0 ± 5.5
Mug-Microwave	27.6 ± 13.5	35.6 ± 10.4	29.2 ± 10.6	43.2 ± 9.9	38.8 ± 5.2	34.4 ± 8.3	39.2 ± 12.7
Libero-Average	21.5	27.72	30.52	36.4	35.42	34.96	**37.76**
			OXE Evaluations				
Franka-Ball	21.4	35.7	28.6	50.0	28.6	57.1	50.0
Franka-Pouch	17.6	17.6	11.8	41.2	35.3	29.4	70.6
Tiago-Sink	33.3	43.6	35.9	46.2	46.2	38.5	64.1
Droid (Drawer)	0.0	0.0	0.0	20.0	70.0	55.0	75.0
Droid (Bread)	4.2	33.3	20.8	0.0	41.7	16.7	41.7
Droid (Napkin)	25.0	45.0	40.0	45.0	40.0	35.0	65.0
Droid-Multitask	9.4	26.6	20.3	20.3	50.0	34.4	59.4
Real-Average	20.4	30.9	24.1	39.4	40.0	39.8	**61.0**

Table 3: Numerical results for LIBERO and OXE evaluations

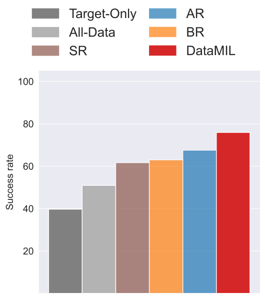


Figure 9: Avg. success on MetaWorld with no-goal conditioning.

MetaWorld. In Figure 3 of the main paper, we presented results on MetaWorld with *goal conditioning*, where policies receive explicit goal information provided by the simulator. Goal-conditioning is often essential in settings like LIBERO and OXE, where the target task has limited demonstrations and generalization from other tasks is required. However, in MetaWorld, the prior dataset already includes expert demonstrations for the target tasks. Therefore, an effective data selection method should be capable of retrieving relevant examples—even in the absence of goal information.

To test this, we repeat the MetaWorld experiments described in Appendix E.1, but mask out goal states during both data selection and policy training. Results averaged over all 50 tasks are shown in Figure 9. Even without goal-conditioning, DataMIL continues to outperform the best baseline by 10% in average success rate. However, overall performance across all methods declines compared to the goal-conditioned setting (Figure 3a), highlighting that, without goal information, selected data from other tasks can introduce harmful interference during training.

LIBERO. Task-wise numerical success rates for the LIBERO and real-world settings are reported in Table 3.

Our results on LIBERO differ from those reported in the original **STRAP** paper on a similar setting, and we attribute these differences to two key factors:

1. **Evaluation Protocol.** We suspect that the main contributing factor is likely a difference in the evaluation protocol. In the original STRAP implementation, the authors evaluate multiple training checkpoints and report results from the best-performing model. This can lead to higher reported success rates. In contrast, our evaluation protocol follows a stricter setup: we evaluate the policy only once, at the final checkpoint after training completes, without any checkpoint selection.
2. **Policy Architecture.** While we use the original STRAP data retrieval code, we differ in the policy architecture used for imitation learning. Specifically, we train with Octo [4], a large

> transformer-based diffusion policy, whereas STRAP uses a transformer-based policy from Robomimic [26] with a Gaussian mixture head. These two models have different inductive biases, which can lead to variation in performance across tasks. For example, when trained solely on the five target demonstrations from the *moka-moka* task, Octo achieves a 6% success rate, while Robomimic achieves 0%. However, in the *mug-mug* task, the Robomimic policy reaches 38% success, while Octo performs close to 0%.

We believe that both protocol differences and model architecture contribute to the gap in reported numbers, and since our goal is to study data selection, our results reflect a fair and consistent evaluation under a unified training and assessment setup across all baselines.

G Limitations

Computational efficiency. Even with the efficient metagradient-based datamodel estimator, the cost of estimating datamodels is several times the cost of training a model on all of the data, which can be prohibitively expensive in many applications. A promising avenue for future work would be to study the extent to which we can accelerate datamodel training, for example by using a scaled-down version of the model of interest or prior dataset.

Hyperparameters. Another limitation of DataMIL (and data selection techniques for robot learning more broadly) is the existence of hyperparameters. In particular, we lack intuition around how to make many design choices (e.g., the target dataset size, clustering hyperparameters, etc.) that affect final performance of DataMIL. On the other hand, such intuition is typically a natural byproduct of methods maturing and being integrated into practice, and so we expect future iterations of DataMIL to get progressively easier to use.

Target task scale. Finally, while we showed data selection from large prior datasets, our target datasets mostly comprised of single tasks. While these tasks show that DataMIL is a promising method for curating training data in the robot learning setting, they may not be the large-scale settings where one expects data curation to be most important. Even though the **Droid-Multitask** setting attempts to simulate this goal, an important direction for future work is to evaluate how DataMIL and other approaches perform on larger-scale target tasks and more expansive prior datasets.