Data-Efficient Multitask DAgger

Haotian Fu[1*], Ran Gong[2], Xiaohan Zhang[2], Maria Vittoria Minniti[2], Jigarkumar Patel[2], Karl Schmeckpeper[2]
[1]Brown University [2]Robotics and AI Institute

***Abstract*— Generalist robot policies that can perform many tasks typically require extensive expert data or simulations for training. In this work, we propose a novel Data-Efficient *multitask DAgger* framework that distills a single multitask policy from multiple task-specific expert policies. Our approach significantly increases the overall task success rate by actively focusing on tasks where the multitask policy underperforms. The core of our method is a performance-aware scheduling strategy that tracks how much each task's learning process benefits from the amount of data, using a Kalman filter-based estimator to robustly decide how to allocate additional demonstrations across tasks. We validate our approach on MetaWorld, as well as a suite of diverse drawer-opening tasks in IsaacLab. The resulting policy attains high performance across all tasks while using substantially fewer expert demonstrations, and the visual policy learned with our method in simulation shows better performance than naive DAgger and Behavior Cloning when transferring zero-shot to a real robot without using real data.**

I. Introduction

Recent progress in robot learning has produced multitask policies [1], [2], [3], [4], [5] capable of performing many manipulation tasks, moving towards the goal of foundation models for robotics. A major challenge in training such multitask policies is the requirement of large and diverse demonstration datasets covering all tasks of interest. Prior works have collected massive human teleoperated demonstration sets or autonomously generated data for dozens of tasks, but at the cost of considerable time and effort. Making multitask imitation learning more data-efficient is crucial for scaling up the number of tasks a single policy can learn.

In imitation learning, Dataset Aggregation (DAgger) [6] is a powerful technique to improve policy performance by iteratively collecting new expert data on states where a learned policy is likely to make mistakes. However, applying DAgger across many tasks raises a key question: how should we distribute the limited budget of data collection queries among the tasks? A naive strategy might allocate data uniformly or equally to each task. This simplistic approach is inefficient because it can waste expert effort on tasks where the policy already performs well, while not providing enough data on harder tasks. Conversely, an ideal strategy would focus on tasks that are currently challenging for the policy and where additional data would yield the greatest improvement.

In this paper, we introduce a *data-efficient multitask DAgger* framework aiming to learn a multitask policy with much less data collection cost. We show the overview of the method in Figure 1. We first train multiple task-specific expert policies (one per task) via reinforcement learning on state-based observations. Then our approach distills these experts into a single vision-based multitask policy (using point cloud or RGB inputs) via iterative imitation learning. Crucially, instead of querying each expert equally, our method monitors the multitask policy's performance on each task and strategically biases data collection towards tasks that need it most. We design a performance-aware scheduler that uses one of two alternative metrics to prioritize tasks: **Task Need (TN)** - defined as the current success rate of the multitask policy on that task measured online during data collection (No additional evaluation phase is required to obtain TN), or **Performance Gain (PG)** - defined as the reduction in the task's behavior cloning loss after the latest training update, which reflects how much the policy's performance on that task improves with new data. We normalize and combine these metrics into a priority score for each task. For the TN metric, we update the success-rate estimates using a Kalman filter to smooth out noise, enabling the scheduler to more reliably select which tasks get additional demonstrations in each DAgger iteration.

To summarize, our key contributions are as follows:

- We propose a novel **data-efficient multitask DAgger framework** that distills a single multitask policy from multiple experts, significantly reducing the amount of expert-provided data needed per task.
- We introduce a **performance-aware data collection strategy** that tracks success rates and learning progress for each task, using a Kalman filter-based estimator to decide how to allocate new expert demonstrations across tasks for maximal performance gain.
- We present **simulation results** on MetaWorld and IsaacSim. The multitask policy outperforms behavior cloning baselines, and our performance-aware data scheduling helps the policy reach higher success rates with less data compared to a uniform DAgger strategy.
- We demonstrate **sim-to-real transfer**: the learned visual multitask policy, trained with our proposed method in simulation, exhibits better real world performance, especially on unseen objects. Robot demos can be found at `https://sites.google.com/brown.edu/mtdagger/home`.

II. Related Work

MultiTask Imitation Learning. Learning generalist policies has been explored in a number of recent works [7], [8], [9], [10], [11], [12], [13], [1], [14], [2], [15], [16]. While

*Work done as an intern at Robotics and AI Institute

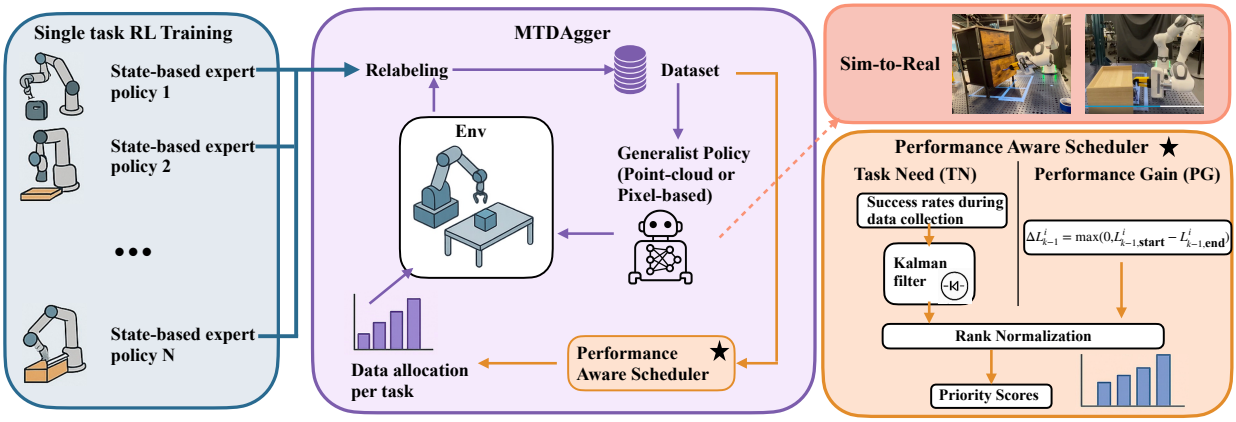


Fig. 1: **Overview of our data-efficient multitask DAgger framework.** Multiple task-specific expert policies (blue) provide demonstrations to train a single multitask policy. In each DAgger iteration, the multitask policy is deployed on all tasks and a *performance-aware scheduler* (orange) uses the Task Need (success rate) and Performance Gain (loss improvement) metrics to prioritize which tasks should receive new expert demonstrations. This focused data collection yields a high-performing generalist policy with far fewer demonstrations. The multitask policy trained in simulation can be directly transferred to real-world tasks.

some of these models were pretrained on image or vision-question answering datasets [12], [11], all of them require some robot-specific data. This data frequently comes from large-scale human teleoperation in the real world [17], [9], [18]. On the other hand, our work follows the footsteps of other multitask policies trained with simulated data [19], [20]. However, rather than uniformly collecting a large number of trajectories, it intelligently samples from a number of expert policies to best improve the policy performance. Training generalist agents that can perform many tasks is a central goal in robotics. Our work also connects to curriculum learning [21], [22], [23].

Interactive Imitation Learning Algorithms. Our approach builds on the DAgger algorithm [6], which established the benefit of iterative expert feedback to correct compounding errors in learned policies. A key limitation of DAgger is the high cost of querying the expert at every timestep. Several works [24], [25] propose to query the expert when the policy is most uncertain. For example, [26], [27] focus on preventing mistakes. Other works have learned failure predictors [26], used disagreement across an ensemble of learners [28], or leveraged model-uncertainty [29] to choose when to query the expert. Like these works, we seek to limit expert intervention; however, instead of a binary query decision for a single task, our framework must decide *which* task's expert to query at each step. To our knowledge, adaptively allocating a demonstration budget across **multiple tasks** within a DAgger-style training process has not been explored previously.

Simulation-to-Real Transfer. Because we train policies in simulation and then evaluate on real robots, our work also relates to the literature on sim-to-real transfer [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42]. Domain randomization techniques [43], [44], [45], [46] have proven effective for learning robust policies that transfer to the real world by training on sufficiently varied simulated observations. More recently, Singh et al. [47] demonstrated zero-shot transfer of a high-dimensional visuomotor policy (for dexterous grasping) from photorealistic simulation to a real robot hand. In the context of multitask learning, Tang et al. [48] report successful zero-shot sim-to-real deployment of both specialist and generalist assembly policies. Inspired by these advances, we show that our generalist policy—trained with data-efficient DAgger in simulation—can likewise perform real-world tasks with minimal degradation, without any real-world fine-tuning.

III. Method

We aim to train a single **multitask visuomotor policy** π_G capable of performing N distinct manipulation tasks, by efficiently distilling knowledge from N pre-trained task-specific expert policies $\{\pi_1^*, \ldots, \pi_N^*\}$. Our core contribution is a data-efficient multitask DAgger algorithm that adaptively allocates expert queries based on the generalist's learning progress and current performance across tasks. Below, we outline the problem setup and then detail our algorithm framework.

A. Problem Formulation

We model each task $i \in \{1, \ldots, N\}$ as a Markov Decision Process (MDP) $\mathcal{M}_i = (\mathcal{S}_i, \mathcal{A}, P_i, r_i, \gamma)$, where $\mathcal{S}_i$ is the state space, $\mathcal{A}$ is the shared action space, P_i is the transition dynamics, r_i is the reward function, and γ is the discount factor. We use reinforcement learning to learn a expert policy $\pi_i^*(a \mid s)$ for each task, typically operating on low-dimensional state $s \in \mathcal{S}_i$.

Our goal is to train a single multitask policy $\pi_G(a \mid o, z_i)$ that maps high-dimensional observations $o \in \mathcal{O}$ (e.g., point clouds or images) and a task identifier z_i (e.g., a one-hot or learned task embedding) to actions $a \in \mathcal{A}$. The multitask policy is trained via imitation learning to mimic the experts'

behavior. Standard DAgger [6] iteratively collects expert labels $a^* = \pi_i^*(s)$ for states s encountered when rolling out the current learner policy $\pi_G^{(k-1)}$ for each round k. The policy is then retrained on the aggregated dataset $D^{(k)} = D^{(k-1)} \cup \{\text{new relabeled data}\}$. In the multitask setting, a key challenge is determining how to distribute the budget for collecting new expert demonstrations $\{(o, a^*, i)\}$ across the N tasks at each iteration k. Naive uniform allocation can be inefficient: it may spend demonstrations on tasks that the policy already handles well, while failing to provide enough data on tasks where the policy struggles. Therefore, we develop a performance-aware scheduling strategy (Section III-D) to intelligently allocate queries to the tasks that need them most.

B. Single-task State-Based Expert Policy Training

Before starting the multitask DAgger pipeline, we first train individual expert policies for each task. Each expert is trained with reinforcement learning algorithms like Proximal Policy Optimization (PPO) [49]. A key aspect of this stage is that these single-task experts operate on low-dimensional state representations of the environment (e.g., proprioceptive information, object poses). Training RL policies using state observations is generally easier than using visual observations and leads to robust expert performance for individual tasks. These state-based experts then provide demonstration data for our primary multitask policy, which, in contrast, is trained to operate directly from high-dimensional sensory inputs (e.g., point clouds or RGB images). This strategy effectively circumvents the challenges of learning complex visuomotor skills from scratch using RL by leveraging the guidance of easily trained state-based experts within an imitation learning framework.

C. Data-Efficient Multitask DAgger

Our approach extends the classic Dataset Aggregation (DAgger) algorithm [6] to the multitask setting by incorporating a performance-aware scheduler for data collection. We maintain a single vision-based *multitask* policy π_G that is trained to perform all N tasks using demonstrations from task-specific *expert* policies $\{\pi_i^*\}_{i=1}^N$. The training process is iterative and interleaves policy improvement with targeted data aggregation (summarized in Algorithm 2). We begin by training an initial policy $\pi_G^{(0)}$ on a small dataset $D^{(0)}$ of expert demonstrations using behavioral cloning (BC) via supervised imitation loss minimization:

$$\pi_G^{(0)} = \arg\min_{\pi_G} \sum_{i=1}^{N} \sum_{(o,a^*,z_i)\in D_i^{(0)}} \mathcal{L}_{\text{BC}}\big(\pi_G(a \mid o, z_i),\, a^*\big), \quad (1)$$

where $\mathcal{L}_{\text{BC}}$ is the imitation loss (e.g., Mean Squared Error for continuous actions). This yields a reasonable starting policy. We also set an initial *DAgger mixing coefficient* $\epsilon_0 \in [0, 1]$, controlling the probability of executing the learned policy versus the expert during data collection; ϵ_0 can be initially small to favor expert actions for safety.

The core of the algorithm iterates for K rounds ($k = 1, \ldots, K$). In each round k, the current multitask policy $\pi_G^{(k)}$ is first refined by training on the aggregated dataset $D^{(k-1)}$ using the BC loss (Eq. 1). Then, the performance-aware scheduler (detailed in Sec. III-D) analyzes task-specific performance metrics (like loss reduction or success rate) to determine an allocation budget n_k^i for collecting new demonstrations for each task i, prioritizing tasks where $\pi_G^{(k)}$ underperforms or shows high learning potential, such that the total demonstrations collected $\sum_i n_k^i$ approximates the round budget B. Following the schedule, new data is collected by deploying $\pi_G^{(k)}$ for n_k^i episodes per task. During these rollouts, actions are chosen by mixing the policy's output and the expert's action based on the previous mixing coefficient ϵ_{k-1}. Crucially, the expert's action a_t^* is always recorded alongside the observation o_t and task identifier z_i. This newly collected data $\Delta D_k^i = \{(o_t, a_t^*, z_i)\}$ for all tasks is aggregated into the main dataset: $D^{(k)} = D^{(k-1)} \cup \bigcup_{i=1}^N \Delta D_k^i$. Finally, the mixing coefficient is decayed (e.g., $\epsilon_k = \max(\epsilon_{k-1} - \Delta\epsilon, \epsilon_{\min})$), gradually shifting control to the learned policy. This iterative process of refining the policy, scheduling data collection based on performance, collecting targeted demonstrations, and aggregating data continues until a stopping criterion is met, resulting in the final policy $\pi_G^{(K)}$.

Algorithm 1: Data-Efficient Multitask DAgger

Require: N tasks, experts π_i^*, initial demos $D^{(0)}$, iterations K, budget B per iteration, initial epsilon ϵ_0, min demos $n_{\min}$, epsilon decay $\Delta\epsilon$.
Train $\pi_G^{(0)}$ on $D^{(0)}$ (Eq. 1). Initialize KF states $\{\hat{p}_0^i, P_0^i\}_{i=1}^N$.
for $k = 1$ to K:

1: **Train Policy:** Update $\pi_G^{(k)}$ by training on $D^{(k-1)}$. Record task losses $L_{k-1,\text{start}}^i, L_{k-1,\text{end}}^i$. Compute gain $g_{k-1}^i = \max(0, L_{k-1,\text{start}}^i - L_{k-1,\text{end}}^i)$.
2: **Schedule Data Allocation:**
- Update Kalman estimates $\hat{p}_{k-1}^i$ using measurements $\bar{p}_{k-1}^i$ from previous DAgger collection (iteration $k-1$) via standard Kalman filter update.
- Compute normalized need $\tilde{d}_{k-1}^i$ (based on $1 - \hat{p}_{k-1}^i$) and gain $\tilde{g}_{k-1}^i$ (based on g_{k-1}^i) using rank normalization.
- Calculate priority score $\text{score}_{k-1}^i = \tilde{g}_{k-1}^i$ or $\tilde{d}_{k-1}^i$ and allocate n_k^i demos per task using softmax over scores.

3: **DAgger Data Collection:** For each task i, collect n_k^i trajectories using $\pi_G^{(k)}$ mixed with π_i^* (with prob ϵ_{k-1}), storing expert labels (o_t, a_t^*, z_i) in ΔD_k^i. Measure raw success rate $\bar{p}_k^i$ for next iteration's scheduling.
4: **Update Dataset and Parameters:** $D^{(k)} \leftarrow D^{(k-1)} \cup \bigcup_i \Delta D_k^i$. Decay $\epsilon_k \leftarrow \max(\epsilon_{k-1} - \Delta\epsilon, \epsilon_{\min})$.

end for
Return $\pi_G^{(K)}$

Fig. 2: Overview of the proposed data-efficient multitask DAgger algorithm.

D. Performance-Aware Scheduling via Adaptive Filtering and Allocation

The scheduler determines the allocation $\{n_k^i\}_{i=1}^N$ of expert demonstrations for the current iteration k based on performance metrics observed up to the previous iteration $(k-1)$. The core idea is to prioritize tasks based on either the

estimated 'Task Need' or the measured 'Performance Gain'. We evaluate these two criteria separately:

a) Task Need (Inverse Success Rate):: We quantify how much a task currently needs improvement by considering the policy's success probability on it. A lower success probability signifies a higher need for corrective expert data. Specifically, we track the filtered success probability estimate $\hat{p}^i_{k-1}$ (detailed below) and use the *inverse success probability*, $1 - \hat{p}^i_{k-1}$, as the metric for measuring TN. The motivation is straightforward: allocate more resources to tasks where the policy is frequently failing, as these represent the largest gaps in performance.

b) Performance Gain (Learning Progress):: Alternatively, we consider how rapidly the policy is improving on each task, measured by the magnitude of the drop in imitation loss within one DAgger iteration. The PG for task i from DAgger iteration $k-1$ is $g^i_{k-1} = \max(0, L^i_{k-1,\text{start}} - L^i_{k-1,\text{end}})$, where $L^i_{k-1,\text{start}}$ describes the loss at the beginning of training for this iteration $k-1$, and $L^i_{k-1,\text{end}}$ describes the loss after a predefined number of training steps. Large PG suggests the task is in a "steep" region of the learning curve, and additional data might yield substantial further improvements efficiently. The motivation here is to capitalize on learning momentum: invest more data where the policy is currently learning effectively, potentially accelerating convergence on those tasks. PG can be particularly useful as an alternative scheduling criterion when initial success rates are near zero for multiple tasks, making TN less discriminative for prioritizing data collection.

To make robust scheduling decisions, we need a stable estimate of each task's success probability. Raw success rates measured during data collection can be noisy, especially when a task is allocated only a few demonstration rollouts in a given round. A short string of lucky or unlucky outcomes can give a misleading impression of the policy's true capability on a task. To filter out this noise, we employ a Kalman filter for each task. The filter treats the true success probability as a hidden state and recursively updates its belief by combining the previous estimate with the new, noisy measurement. This provides a principled way to weigh new evidence against our prior belief, trusting measurements from many rollouts more than those from few. This smoothed estimate, $\hat{p}^i$, provides a more reliable signal for our 'Task Need' (TN) scheduling metric.

Empirically, we found that directly using the two metrics without normalization can lead to some overly tilted data distribution. For example, if one hard task got 0 success rate and the others got over 50% success rate, almost all the data are collected from that hard task for the next iteration. To ensure robustness regardless of the chosen scheduling metric ('need' derived from $1-\hat{p}^i_{k-1}$ or 'gain' g^i_{k-1}), which may have different scales and distributions, we employ rank normalization. For the selected metric m (either need or gain), let $\{m^1, \ldots, m^N\}$ be the values across the N tasks. Let $\text{rank}(m^i)$ denote the rank of task i based on this metric (e.g., rank 1 for lowest value, rank N for highest value). The rank-normalized value $\tilde{m}^i$ is computed as:

$$\tilde{m}^i = \frac{\text{rank}(m^i) - 1}{N - 1}, \quad \text{for } N > 1 \quad (\text{and } \tilde{m}^i = 0 \text{ if } N = 1). \tag{2}$$

This scales the ranks to the range $[0, 1]$, preserving the relative ordering while making the metrics scale-invariant and robust to outliers. We compute the normalized need $\tilde{d}^i_{k-1}$ (ranking $1-\hat{p}^i_{k-1}$) and normalized gain $\tilde{g}^i_{k-1}$ (ranking g^i_{k-1}) using this method.

We then take the outputs from normalization and use them as a priority score for each task i. We convert these scores into allocation probabilities using softmax and decide the number of trajectories allocated for each task by taking the product of this probability and the budget for the total number of trajectories to be allocated at each round. This adaptive mechanism dynamically focuses expert queries on tasks that are most likely to benefit, driving data efficiency.

IV. Experiments

In this section, we evaluate our MultiTask DAgger algorithm, designed to enhance sample efficiency in multitask imitation learning. We conduct experiments in two distinct domains: the standard Meta-World benchmark, and a set of drawer-opening tasks in IsaacLab [50]. Our goals are to demonstrate: (1) Improved sample efficiency and asymptotic performance compared to standard Behavior Cloning (BC) and uniform DAgger approaches. (2) The effectiveness of our performance-aware scheduling mechanism. (3) The effectiveness of the trained multitask policy in real-world hardware experiments through **zero-shot sim-to-real transfer**.

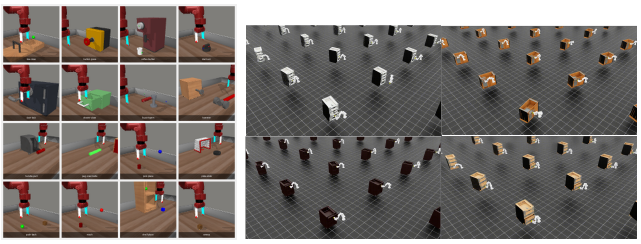


Fig. 3: Examples of Meta-World tasks and IsaacLab Drawer tasks used in our experiments.

A. Meta-World Benchmark and Drawer-Opening in IsaacLab

To evaluate scalability and performance on a standard benchmark, we utilize the MetaWorld [51] suite, specifically the Multitask setup where we select 36 distinct manipulation tasks. We first train a state-based Soft-Actor-Critic (SAC) [52] expert policy for each task. Specifically, we apply the SAC algorithm to all 50 tasks comprising the benchmark, and success rates surpass 80% on 36 of these tasks. Then we train multitask policies using our method under two conditions: 1. **State-based Multitask Policy:** The multitask policy receives low-dimensional state observations, mirroring the expert's input space but requiring generalization across tasks. 2. **Image-based Multitask Policy:** The

multitask policy receives RGB image observations, requiring learning from high-dimensional visual input in addition to multitask generalization. This setup allows us to compare performance and sample efficiency against baselines under varying input modalities and task complexities. To assess generalization across geometrically diverse but functionally similar objects, we use an IsaacLab environment featuring a Franka robot arm. The task involves opening various drawers. The training task suite includes 11 different drawers. 10 of them are sourced from the PartNet-Mobility [53], [54] dataset, selected for their diverse geometries and articulation mechanisms, plus one which models a real drawer that we have. Each task requires learning a policy to successfully open its specific drawer. Single-task expert policies are trained using PPO [49] on low-dimensional state representations. Our goal is to train a single, multitask policy using our adaptive DAgger approach that operates directly from high-dimensional camera observations and can successfully operate all 11 drawers. A visualization of the simulated setup with the standard drawer is shown in Figure 3 right.

B. Setup

For the Meta-World state-based experiments, for each DAgger round, we collect 108 demonstrations in total per round across 36 tasks, starting with 3 initial demonstrations per task. For the Meta-World pixel-based experiments, for each round, we collect 720 demonstrations in total per round across 36 tasks, starting with 40 initial demonstrations per task. For the Isaac Drawer experiments, we ran for 10 rounds, collecting 30 demonstrations per task per round across 11 tasks.

We compare our Data-efficient Multitask DAgger against two primary baselines: 1. **BC** A standard imitation learning approach where a policy is trained offline on a fixed dataset of expert demonstrations collected uniformly across all tasks. 2. **Uniform DAgger:** The standard DAgger algorithm where, in each round, an equal number of demonstrations are collected for every task, regardless of estimated difficulty or learning progress. The primary metric for evaluation is the average success rate across all tasks in the respective benchmark. Success rates are measured by deploying the current multitask policy checkpoint for a fixed number of episodes per task and calculating the percentage of successful completions. We report these average success rates throughout the training process (across DAgger rounds) to analyze sample efficiency.

C. Simulation Learning Performance Comparison

Figure 4 presents the learning curves comparing the average success rates of our proposed multitask DAgger variants (MTDAgger-PG and MTDAgger-TN) against BC with varying data budgets and standard Uniform DAgger. Across all three experimental settings – MetaWorld state-based (left), MetaWorld pixel-based (middle), and IsaacLab point-cloud-based (right) – our performance-aware scheduling strategies consistently demonstrate superior sample efficiency and achieve higher final performance. Notably, MTDAgger-PG generally performs better than MTDAgger-TN, showing that the success rate is generally a more stable metric for calculating the task difficulty, even though it is calculated during data collection. This efficiency gain is further quantified in Figure 5, which shows the substantial reduction in the total number of expert trajectories required by our methods to reach target success rates (e.g., 60% and 80% in MetaWorld, 52% in IsaacLab) compared to one-round BC and Uniform MTDAgger. These results validate the effectiveness of adaptively allocating data collection efforts based on TN or PG, leading to more data-efficient learning of multitask policies from multiple experts across diverse tasks and observation modalities.

D. Analysis of Adaptive Scheduling

A key component of our method is the ability to estimate task needs and allocate resources accordingly. We verify the effectiveness of our scheduling metrics. Figure 8 first plot shows the correlation between the estimated TN (derived from the Kalman filter's success rate estimate) in the final rounds and the actual measured success rate during evaluation. Figure 8 second plot similarly shows the correlation for the PG metric (ΔL_i). The results show that for those tasks where the evaluation success rate is high, our metrics give it lower scores which means it will be scheduled to collect fewer trajectories next round, and vice versa. Both plots indicate a meaningful correlation, suggesting that our metrics effectively capture aspects of task difficulty and learning potential, thereby justifying their use in guiding the data allocation process.

Figures 8 right shows the learning curves for two of the most challenging tasks in the Meta-World setting, 'dial-turn-v2' and 'sweep-into-v2', respectively. We run BC with 1000 demonstrations per task for all 36 tasks and find these two are the hardest tasks (lowest success rates). Compared to the plots for the comparison of success rate averaged across all the tasks, we see on these hard tasks our methods demonstrate even better performance with a larger gap compared to Uniform DAgger. PG Metric shows better performance than TN Metric on these tasks, indicating that prioritizing tasks by learning progress can be especially effective for extremely challenging tasks.

Figure 7 presents supplementary results further analyzing the performance and characteristics of our proposed method. Figure 7(a) provides an ablation study on the Meta-World state-based tasks, demonstrating that incorporating the Kalman filter (MTDAgger-TN) yields slightly better final performance and stability compared to using raw success rates for the Task Need scheduler (TN without Kalman filter), validating the benefit of smoothed success rate estimation. Figure 7(b) compares the wall-clock training time required to reach target success rates (60% and 80%) on MetaWorld pixel-based tasks, highlighting that MTDAgger-TN achieves these thresholds significantly faster than standard one-round Behavior Cloning, owing to its data efficiency.

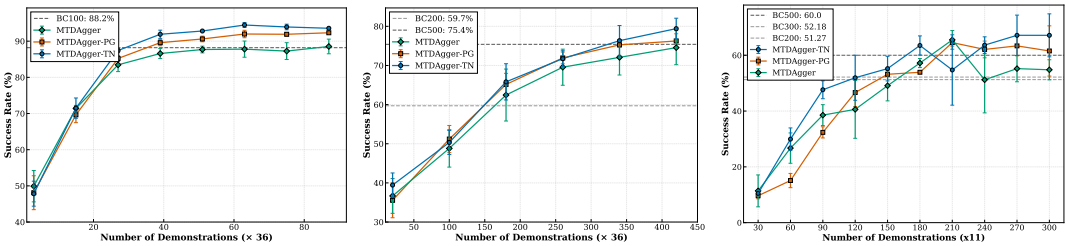


Fig. 4: Average success rate vs. the number of expert demonstrations collected per task (averaged). Left: Metaworld state-based multitask policy comparison. Middle: Metaworld pixel-based multitask policy comparison. Right: IsaacLab Drawer point-cloud-based multitask policy comparison. Our performance-aware scheduling mechanism generally achieves higher sample efficiency and final performance compared to standard BC with varying data budgets and Uniform DAgger.

TABLE I: Sim-to-Real Performance on in-domain vs. out-of-domain tasks. (15 trials each)

Method	In-Domain (Success Rates)	Unseen (Success Rates)	In-Domain (Task Progress)	Unseen (Task Progress)
Ours	$\mathbf{0.60 \pm 0.28}$	$\mathbf{0.53 \pm 0.25}$	$\mathbf{2.40 \pm 0.57}$	$\mathbf{2.27 \pm 0.75}$
MTDAgger	0.33 ± 0.25	0.13 ± 0.19	1.80 ± 0.57	0.67 ± 0.52
BC	0.40 ± 0.33	0.13 ± 0.09	1.20 ± 0.98	0.40 ± 0.28

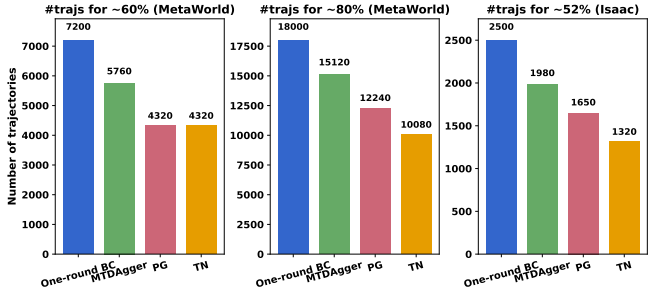
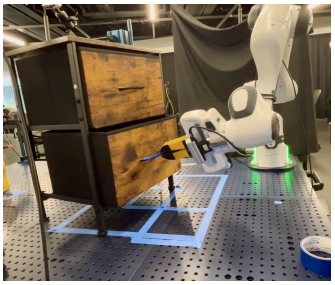
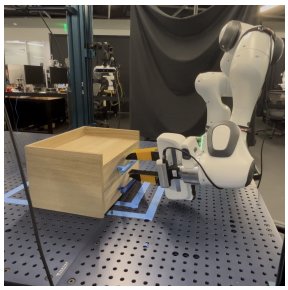


Fig. 5: Comparison of number of trajectories needed to reach high success rates in MetaWorld and IsaacLab Drawer tasks.



(a) Unseen Drawer

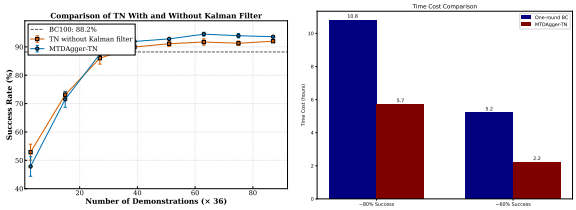


(b) In-Domain Drawer

Fig. 6: Images of the real-world Franka setup for the drawer opening task.

E. Sim-to-Real Evaluation

To assess the practical applicability of policies trained with our framework, we performed sim-to-real transfer experiments. For perception in the real world, the robot uses a Intel RealSense D435 camera. The raw point cloud data from the camera was cropped to remove background with 4096 points uniformly sampled. As shown in Figure 6, we tested on a domain drawer that is included in the simulation training set, as well as an unseen drawer that was not in the training data. We directly use the multitask policy we trained in simulation to rollout on the real robot, for three different drawer positions, 5 trials each. We measure the success rates on these two drawer for different methods, as well as the task progress, where the robot will get 1 point each for grasping the handle, attempting to pull, and opening the drawer fully (max 3 points). Due to the greater difficulty presented by the unseen drawer task, a less stringent success criterion was established, requiring only partial opening of the drawer for the attempt to be considered successful.



(a) (b)

Fig. 7: Addtional experiment results. (a) Performance if we do not use Kalman filter for MTDAgger-TN. (b) Wall time cost for training comparison on MetaWorld.

F. Robot Demos

We show videos of real robot rollouts on in-domain and unseen drawers respectively[1]. It also contains 2 failure videos which still shows some interesting behavior the robot learned. In the **Failure1_seen_drawer** part, the robot was able to grasp the handle but then it slipped away when the gripper tried to pull the handle. But then the robot shows some interesting **recovery behavior** that it was able to find and grasp the handle again but just not further pull it our. All

[1]https://sites.google.com/brown.edu/mtdagger/home

the videos are shown in the **original speed** and we executed the learned policy on real robot with 10 Hz frequency.

G. Other Metrics we have tried in the experiments

Besides the two metrics – Task Need and Performance Gain, that we have found to be useful in the experiments, in practice we have tried a few other metrics that didn't end up working well. These include:

- KL divergence between the output distribution of the multitask policy and the expert policies
- Normalized final/initial loss of each task during each training round
- Variance of the normalized loss of each task during each training round

V. Conclusion

We presented a framework for efficiently training a multi-task robot policy by integrating DAgger with a performance-aware task scheduler. Our approach significantly reduces the total amount of data required by prioritizing demonstrations for tasks that are either currently difficult or promise high learning gains. We showed that this strategy leads to faster learning and higher final success on complex multitask benchmarks, and enables better zero-shot transfer of the learned policy to real robotic tasks including unseen tasks.

Acknowledgement

The authors would like to thank Lingfeng Sun, Xiaoqiang Yan, Jinghuan Shang, Laura Herlant, George Konidaris for helpful discussions and feedbacks.

References

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π0: A vision-language-action flow model for general robot control, 2024," *URL https://arxiv.org/abs/2410.24164*.

[2] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai *et al.*, "π0.5: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[3] 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," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "Openvla: An open-source vision-language-action model," in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270. PMLR, 06–09 Nov 2025, pp. 2679–2713.

[5] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 576–11 582.

[6] S. Ross, G. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 627–635.

[7] A. Singh, E. Jang, A. Irpan, D. Kappler, M. Dalal, S. Levinev, M. Khansari, and C. Finn, "Scalable multi-task imitation learning with autonomous improvement," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2167–2173.

[8] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[9] 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 (CoRL)*, 2022, pp. 991–1002.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *arXiv:2212.06817*, 2022, available at arXiv:2212.06817.

[11] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, B. Ichter, A. Zeng *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning (CoRL)*, 2023.

[12] D. Driess, F. Xia, M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter *et al.*, "PaLM-E: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[13] S. Haldar, Z. Peng, and L. Pinto, "Baku: An efficient transformer for multi-task policy learning," *arXiv preprint arXiv:2406.07539*, 2024.

[14] H. Fu, P. Sharma, E. Stengel-Eskin, G. Konidaris, N. L. Roux, M. Côté, and X. Yuan, "Language-guided skill learning with temporal variational inference," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[15] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[16] W. Wan, Y. Zhu, R. Shah, and Y. Zhu, "Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 537–544.

[17] 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.

[18] 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)*. IEEE, 2024, pp. 6892–6903.

[19] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," *arXiv preprint arXiv:2210.03094*, vol. 2, no. 3, p. 6, 2022.

[20] 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.

[21] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, ser. ACM International Conference Proceeding Series, A. P. Danyluk, L. Bottou, and M. L. Littman, Eds., vol. 382. ACM, 2009, pp. 41–48.

[22] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 166–175.

[23] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an RL agent using LTL," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, E. André, S. Koenig, M. Dastani, and G. Sukthankar, Eds. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 452–461.

[24] K. Menda, K. R. Driggs-Campbell, and M. J. Kochenderfer, "Ensembledagger: A bayesian approach to safe imitation learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*. IEEE, 2019, pp. 5041–5048.

[25] X. Zhang, M. Chang, P. Kumar, and S. Gupta, "Diffusion meets dagger: Supercharging eye-in-hand imitation learning," in *Robotics:*

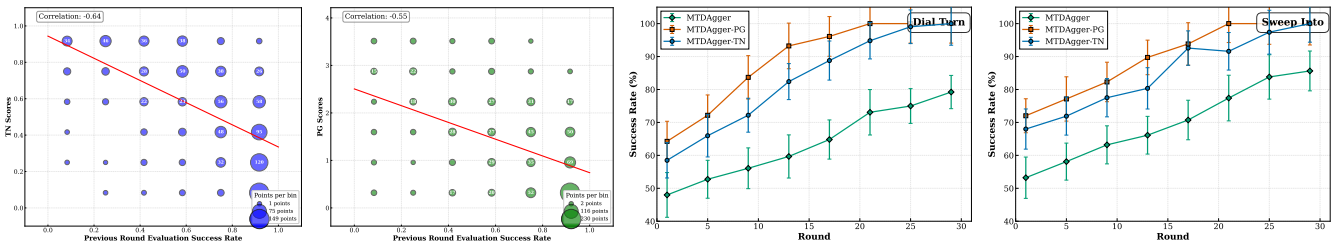


Fig. 8: Left: correlation between the estimated TN metric (based on KF success rate estimate) or PG metric from the final training rounds and the final evaluation success rate across Meta-World tasks. Right: Success rate comparison on 'dial-turn-v2' task and 'sweep-into-v2' task.

Science and Systems XX, Delft, The Netherlands, July 15-19, 2024, D. Kulic, G. Venture, K. E. Bekris, and E. Coronado, Eds., 2024.

[26] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," in *Workshops at the 30th AAAI Conference on Artificial Intelligence*, 2016.

[27] Y. Korkmaz and E. Biyik, "MILE: model-based intervention learning," *CoRR*, vol. abs/2502.13519, 2025.

[28] A. Haridas, Z. Wang, A. Garg, and D. Hsu, "Dagger with ensemble-based error awareness (dadagger)," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8722–8729.

[29] M. Kelly, C. Sidrane, D. Losey, and A. D. Dragan, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8077–8083.

[30] F. Sadeghi and S. Levine, "CAD2RL: real single-image flight without a single real image," in *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, Eds., 2017.

[31] A. Byravan, J. Humplik, L. Hasenclever, A. Brussee, F. Nori, T. Haarnoja, B. Moran, S. Bohez, F. Sadeghi, B. Vujatovic, and N. Heess, "Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields," in *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*. IEEE, 2023, pp. 9362–9369.

[32] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning, 8-11 November 2021, London, UK*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 2021, pp. 91–100.

[33] A. Yu, A. Foote, R. Mooney, and R. Martín-Martín, "Natural language can help bridge the sim2real gap," in *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*, D. Kulic, G. Venture, K. E. Bekris, and E. Coronado, Eds., 2024.

[34] P. Xie, R. Chen, S. Chen, Y. Qin, F. Xiang, T. Sun, J. Xu, G. Wang, and H. Su, "Part-guided 3d RL for sim2real articulated object manipulation," *IEEE Robotics Autom. Lett.*, vol. 8, no. 11, pp. 7178–7185, 2023.

[35] R. Gong, J. Huang, Y. Zhao, H. Geng, X. Gao, Q. Wu, W. Ai, Z. Zhou, D. Terzopoulos, S.-C. Zhu *et al.*, "Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[36] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[37] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka, "Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning," in *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 2023, pp. 1621–1639.

[38] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, "TRANSIC: sim-to-real policy transfer by learning from online correction," in *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 2024, pp. 1691–1729.

[39] J. Bagajo, C. Schwarke, V. Klemm, I. Georgiev, J. Sleiman, J. Tordesillas, A. Garg, and M. Hutter, "Diffsim2real: Deploying quadrupedal locomotion policies purely trained in differentiable simulation," *CoRR*, vol. abs/2411.02189, 2024.

[40] T. G. W. Lum, A. H. Li, P. Culbertson, K. Srinivasan, A. D. Ames, M. Schwager, and J. Bohg, "Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer," in *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 2024, pp. 5405–5433.

[41] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. S. Narang, "Industreal: Transferring contact-rich assembly tasks from simulation to reality," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, Eds., 2023.

[42] E. Su, C. Jia, Y. Qin, W. Zhou, A. Macaluso, B. Huang, and X. Wang, "Sim2real manipulation on unknown objects with tactile-based reinforcement learning," in *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*. IEEE, 2024, pp. 9234–9241.

[43] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[44] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," in *Robotics: Science and Systems (RSS)*, 2017.

[45] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs, "Poliformer: Scaling on-policy rl with transformers results in masterful navigators," *CoRL*, 2024.

[46] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5977–5984.

[47] R. Singh, Z. Huang, Z. Wang, Y. Zhu, K. Daniilidis, and J. Lee, "Splatsim: Zero-shot sim-to-real transfer of rgb-based manipulation policies via rendering with gaussian splatting," *arXiv preprint arXiv:2309.10161*, 2023.

[48] Y. Tang, R. Kulkarni, S. Javdani, S. Srinivasa, and A. Mandlekar, "A framework for teaching diverse assembly tasks to a generalist robot via multimodal simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[50] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[51] 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 *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, ser. Proceedings of Machine Learning Research, L. P.

Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 2019, pp. 1094–1100.
[52] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 1856–1865.
[53] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
[54] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A simulated part-based interactive environment," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

VI. Limitations

While our proposed data-efficient multitask DAgger framework demonstrates significant improvements in sample efficiency and sim-to-real transfer, it possesses several limitations.

Firstly, the method's effectiveness hinges on the availability and quality of pre-trained, task-specific expert policies. Although these experts are trained on state representations for efficiency, the initial cost associated with training N distinct experts via reinforcement learning can still be considerable, particularly as the number of tasks N grows very large. Our approach optimizes the distillation process but does not eliminate this prerequisite expert training phase.

Secondly, although our method achieved better zero-shot sim-to-real transfer compared to baselines, a performance gap between simulation and reality persists, especially for the out-of-domain task. This suggests that challenges related to domain shift are mitigated but not fully resolved by the adaptively sampled data distribution alone, and further techniques like domain randomization or incorporating minimal real-world data might be necessary for higher-fidelity transfer.

Finally, the current implementation has been validated on benchmarks with up to 36 tasks. While the scheduling mechanism is designed to be scalable, the practical overhead of managing a very large number of experts and the computational cost of the scheduling step itself might become factors when scaling to hundreds or thousands of diverse tasks.

VII. Algorithm Details

The process iterates through the following steps (see Algorithm 2):

Initialization: Train an initial policy $\pi_G^{(0)}$ using standard behavioral cloning (BC) on a small initial dataset $D^{(0)} = \bigcup_{i=1}^{N} D_i^{(0)}$ containing a few expert demonstrations for each task:

$$\pi_G^{(0)} = \arg\min_{\pi_G} \sum_{i=1}^{N} \sum_{(o,a^*,z_i)\in D_i^{(0)}} \mathcal{L}_{\text{BC}}\big(\pi_G(a \mid o, z_i),\, a^*\big), \quad (3)$$

where $\mathcal{L}_{\text{BC}}$ is the imitation loss (e.g., Mean Squared Error for continuous actions). Set the initial DAgger mixing coefficient ϵ_0 (probability of using the learner's action during data collection). Initialize Kalman filter states (belief and uncertainty about success probability) $\{\hat{p}_0^i, P_0^i\}_{i=1}^N$ for each task (e.g., $\hat{p}_0^i = 0.5, P_0^i = 0.25$).

Iteration $k = 1, \ldots, K$**:**

1) **Train Policy:** Update the generalist policy $\pi_G^{(k)}$ by training on the current aggregated dataset $D^{(k-1)}$ using Eq. (3). During training, record the initial ($L^i_{k-1,\text{start}}$) and final ($L^i_{k-1,\text{end}}$) imitation loss specific to each task i on a validation set or the training data itself. Calculate the performance gain $g^i_{k-1} = \max(0, L^i_{k-1,\text{start}} - L^i_{k-1,\text{end}})$.
2) **Schedule Data Allocation:** Based on metrics from the *previous* iteration's data collection (iteration $k-1$) and the training process just completed, determine the number of new expert trajectories n_k^i to collect for each task i in the current iteration k. This crucial step uses the performance-aware scheduler detailed in Section III-D, ensuring $\sum_i n_k^i \approx B$, where B is the total demonstration budget per iteration.
3) **DAgger Data Collection:** For each task i, execute $\pi_G^{(k)}$ in the simulator for n_k^i episodes. During these rollouts, with probability ϵ_{k-1}, use the action $a_t = \pi_G^{(k)}(o_t, z_i)$; otherwise use the expert action $a_t^* = \pi_i^*(s_t)$. Critically, *always record the expert label* a_t^* paired with the observation o_t. Collect the resulting demonstration tuples (o_t, a_t^*, z_i) into a temporary set ΔD_k^i. Also, measure the raw success rate $\bar{p}_k^i$ of $\pi_G^{(k)}$ during these n_k^i rollouts (or a fixed number of evaluation rollouts if $n_k^i = 0$). This $\bar{p}_k^i$ will be used for scheduling in the *next* iteration $(k+1)$.
4) **Update Dataset and Parameters:** Aggregate the newly collected data: $D^{(k)} = D^{(k-1)} \cup \bigcup_{i=1}^{N} \Delta D_k^i$. Decay the DAgger mixing coefficient: $\epsilon_k = \max(\epsilon_{k-1} - \Delta\epsilon, \epsilon_{\min})$.

This loop continues until a maximum number of iterations K is reached or a performance criterion is met. The key innovation lies in Step 2, the scheduler, which we detail next.

VIII. Two important techniques of Performance-aware Scheduling

a) Adaptive Kalman Filtering of Success Probabilities.: Raw success probabilities $\bar{p}_{k-1}^i$ measured during DAgger collection can be unreliable for scheduling. The stochasticity of the policy and environment introduces noise, and estimates from a small number of rollouts n_{k-1}^i suffer from high variance. To obtain a more stable estimate of the true success probability, $\hat{p}^i$, we use a Kalman filter for each task i.

The filter recursively updates its estimate $\hat{p}^i$ and associated uncertainty P^i. The process involves two steps:

1) **Predict:** We assume the true success rate is stable between iterations, so the predicted probability is the last estimate, $\hat{p}^i_{k-1|k-2} = \hat{p}^i_{k-2|k-2}$. Our uncertainty in this prediction grows by a small process noise Q: $P^i_{k-1|k-2} = P^i_{k-2|k-2} + Q$.
2) **Update:** The prediction is corrected using the new measurement $\bar{p}^i_{k-1}$. The degree of correction is determined by the Kalman gain K^i_{k-1}, which balances the

predicted uncertainty against the measurement noise R^i_{k-1}. We make the measurement noise adaptive: $R^i_{k-1} = R_0/(n^i_{k-1}+1)$, reflecting higher confidence in estimates derived from more data. A lower measurement noise (more data) leads to a higher Kalman gain, placing more trust in the new measurement.

The standard update equations are then applied:

$$K^i_{k-1} = \frac{P^i_{k-1|k-2}}{P^i_{k-1|k-2} + R^i_{k-1}} \tag{4}$$

$$\hat{p}^i_{k-1|k-1} = \hat{p}^i_{k-1|k-2} + K^i_{k-1}\left(\bar{p}^i_{k-1} - \hat{p}^i_{k-1|k-2}\right) \tag{5}$$

$$P^i_{k-1|k-1} = \left(1 - K^i_{k-1}\right) P^i_{k-1|k-2} \tag{6}$$

The filter's output, $\hat{p}^i_{k-1} = \hat{p}^i_{k-1|k-1}$, provides a smoothed success probability for scheduling, mitigating the effects of noise and high-variance estimates. While the standard Kalman filter assumes Gaussian noise, and success rates are binomial, it serves as a simple and effective approximation for our purposes.

b) Rank Normalization of Performance Metrics.: Empirically, we found that directly using the two metrics without normalization can lead to some overly tilted data distribution. For example, if one hard task got 0 success rate and the others got over 50% success rate, almost all the data are collected from that hard task for the next iteration. To ensure robustness regardless of the chosen scheduling metric ('need' derived from $1-\hat{p}^i_{k-1}$ or 'gain' g^i_{k-1}), which may have different scales and distributions, we employ rank normalization. For the selected metric m (either need or gain), let $\{m^1, \ldots, m^N\}$ be the values across the N tasks. Let $\text{rank}(m^i)$ denote the rank of task i based on this metric (e.g., rank 1 for lowest value, rank N for highest value). The rank-normalized value $\tilde{m}^i$ is computed as:

$$\tilde{m}^i = \frac{\text{rank}(m^i) - 1}{N - 1}, \quad \text{for } N > 1 \quad (\text{and } \tilde{m}^i = 0 \text{ if } N = 1). \tag{7}$$

This scales the ranks to the range $[0, 1]$, preserving the relative ordering while making the metrics scale-invariant and robust to outliers. We compute the normalized need $\tilde{d}^i_{k-1}$ (ranking $1-\hat{p}^i_{k-1}$) and normalized gain $\tilde{g}^i_{k-1}$ (ranking g^i_{k-1}) using this method.

We then take the outputs from normalization and use them as a priority score for each task i. We convert these scores into allocation probabilities using softmax and decide the number of trajectories allocated for each task by taking the product of this probability and the budget for the total number of trajectories to be allocated at each round. This adaptive mechanism dynamically focuses expert queries on tasks that are most likely to benefit, driving data efficiency.

IX. Implementation details

Our method, which we refer to as MTDAgger (-PG or -TN), extends the standard DAgger framework by incorporating a performance-aware data collection schedule. As detailed in Section III, we use a Kalman filter to estimate the success rate (p_i) for each task i, or use performance gain from training, ΔL_i, that informs a scoring function. We use rank normalization on these metrics to calculate task scores. These scores determine the allocation of data collection budget for the next DAgger round using a temperature-controlled softmax function, prioritizing tasks estimated to benefit most from additional expert demonstrations. Key parameters for the scheduling mechanism (e.g., KF process/measurement variance, scoring weights α, γ, softmax temperature) were set based on preliminary experiments and are detailed in the Appendix. The DAgger mixing parameter ϵ was annealed from 0.5 towards 0 over the rounds to transition from expert to learner actions.

We convert the normalized score into allocation probabilities using a temperature-controlled softmax:

$$\rho^i_k = \frac{\exp(\text{score}^i_{k-1}/T)}{\sum_{j=1}^N \exp(\text{score}^j_{k-1}/T)}, \tag{8}$$

where $T > 0$ is the temperature hyperparameter controlling the sharpness of the allocation, and ρ^i_k represents the probability mass allocated to task i in iteration k. The number of demonstrations allocated to task i for the current iteration k is then:

$$n^i_k = \max\left(n_{\min}, \rho^i_k \cdot B\right),$$

with minor adjustments to ensure the total allocation $\sum_i n^i_k$ approximately equals the budget B. We enforce a minimum allocation $n_{\min} \geq 1$ for tasks with non-zero scores. This adaptive mechanism dynamically focuses expert queries on tasks that are most likely to benefit, driving data efficiency.

A. Multitask Policy Architecture for IsaacLab Drawer-opening

To handle diverse tasks within a single network, we employ a policy architecture that conditions on task identity. Each task i is represented by a learned task embedding $z_i \in \mathbb{R}^d$, obtained by passing a one-hot task index through a trainable embedding layer. The policy $\pi_G(a \mid o, z_i)$ takes the current observation o (e.g., robot proprioception and point cloud features) and the task embedding z_i as input.

Our network consists of:

1) An observation encoder $f_{\text{enc}}(o)$ (e.g., a PointNet-based module for point clouds, potentially combined with an MLP for proprioceptive state) that extracts a latent representation h_o.
2) A fusion module that combines the observation representation h_o with the task embedding z_i, for example, by concatenation: $h_{\text{fused}} = [h_o, z_i]$.
3) A policy head $f_{\text{policy}}(h_{\text{fused}})$ (typically an MLP) that outputs the action a.

The shared encoder f_{enc} learns features relevant across all tasks, while the task embedding z_i allows the policy head f_{policy} to specialize its behavior for the specific requirements of task i. This architecture balances generalization and task-specific adaptation effectively for the visuomotor control tasks considered.

Hyperparameters	Batch size	epsilon	epsilon decay	min epsilon	Q	R_0	learning rate	T	n_{min}
MetaWorld	1024	0.5	0.5	0	0.03	0.5	3e-4	0.5	1
IsaacLab	256	0.5	0.5	0	0.03	0.5	1e-4	0.5	5

TABLE II: Hyperparameters we used in our experiments. Although the benchmarks are completely different, the algorithm-related hyperparameters are basically the same.

X. Visualization of the simulated drawers

Fig. 9: Visualization of the 11 drawers we trained on in the experiments.

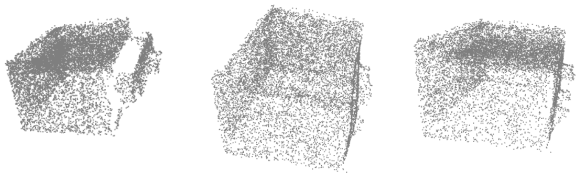


Fig. 10: Examples of the point-cloud observations for real robot experiments.

XI. Other Metrics we have tried in the experiments

Besides the two metrics – Task Need and Performance Gain, that we have found to be useful in the experiments, in practice we have tried a few other metrics that didn't end up working well. These include:

- KL divergence between the output distribution of the multitask policy and the expert policies
- Normalized final/initial loss of each task during each training round
- Variance of the normalized loss of each task during each training round