

A Careful Examination of Large Behavior Models for Multitask Dexterous Manipulation

TRI LBM Team

Abstract—Robot manipulation has seen tremendous progress in recent years, with imitation learning policies enabling successful performance of dexterous and hard-to-model tasks. Concurrently, scaling data and model size has led to the development of capable language and vision foundation models, motivating large-scale efforts to create general-purpose robot foundation models. While these models have garnered significant enthusiasm and investment, meaningful evaluation of real-world performance remains a challenge, limiting both the pace of development and inhibiting a nuanced understanding of current capabilities.

In this paper, we rigorously evaluate multitask robot manipulation policies, referred to as Large Behavior Models (LBMs), by extending the Diffusion Policy paradigm across a corpus of simulated and real-world robot data. We propose and validate an evaluation pipeline to rigorously analyze the capabilities of these models with statistical confidence. We compare against single-task baselines through blind, randomized trials in a controlled setting, using both simulation and real-world experiments. We find that multitask pretraining makes the policies more successful and robust, and enables teaching complex new tasks more quickly, using a fraction of the data when compared to single-task baselines. Moreover, performance predictably increases as pretraining scale and diversity grows. Project page: <https://toyotaresearchinstitute.github.io/lbm1/>

I. INTRODUCTION

Achieving flexible, generalist robots is a central ambition of robotics research. While modern robots are physically capable of performing a wide array of tasks in myriad settings, reliable autonomy has traditionally been limited to simple tasks or highly structured environments. Recently, visuomotor learning-based methods—trained to condition on robot sensor observations and produce low-level actions—have emerged as promising solutions to bridge this gap between hardware capabilities and autonomous performance. Behavior cloning, the most commonly used approach in this regime, eschews task-specific robot programming in favor of task-specific demonstrations, typically collected via teleoperation. Methods based on behavior cloning [1]–[3] can produce complex, reactive, and contact-rich behaviors from hundreds to thousands of demonstrations, are well suited to handle traditionally challenging task attributes such as object deformability, transparency, reflectivity, and bimanual

see Section VI for full author list.

coordination, and offer the promise of producing general-purpose manipulation systems capable of performing arbitrary tasks.

Despite these strengths, single-task behavior-cloned policies remain brittle, exhibiting limited generalization to task variations or environments outside their training distributions. To overcome this brittleness, the field is increasingly adopting *Large Behavior Models* (LBMs) [2], [4]–[9]—visuomotor foundation models trained on large-scale multitask datasets containing action-level demonstrations. Inspired by the success of large-scale generalist models in Computer Vision [10]–[13] and Natural Language Processing [14], [15], these models seek to improve reliability through broad training data support and more robust learned visual and sensory representations. Despite the surge in LBM research and development, significant uncertainty remains regarding the extent to which observed successes primarily stem from multitask pretraining.

To rigorously study the impact of multitask pretraining, we train multiple LBMs on approximately 1,700 hours of robot demonstrations comprised of over 500 internally collected high-diversity tasks as well as publicly available robot data. We comprehensively evaluate these models both in simulation and through 1,800 rigorously controlled real-world trials, including complex multi-step tasks that require tool use and precise manipulation. We design and employ an experimental protocol that incorporates blind A/B testing in the real world, large trial sizes with robust statistical analysis, qualitative and quantitative performance metrics, and carefully controlled initial conditions to ensure our conclusions hold with statistical significance.

Through these experiments, we find that:

- 1) LBM pretraining reduces the amount of task-specific data required, enabling finetuned specialist models to match single-task model performance with fewer demonstrations.
- 2) Given the same amount of task-specific data, finetuned specialists derived from pretrained LBMs outperform single-task models when aggregating over tasks.
- 3) Pretrained LBMs demonstrate increased robustness in scenarios diverging from their training conditions, amplifying the benefits described in points 1 and 2 in out-of-distribution evaluation settings.

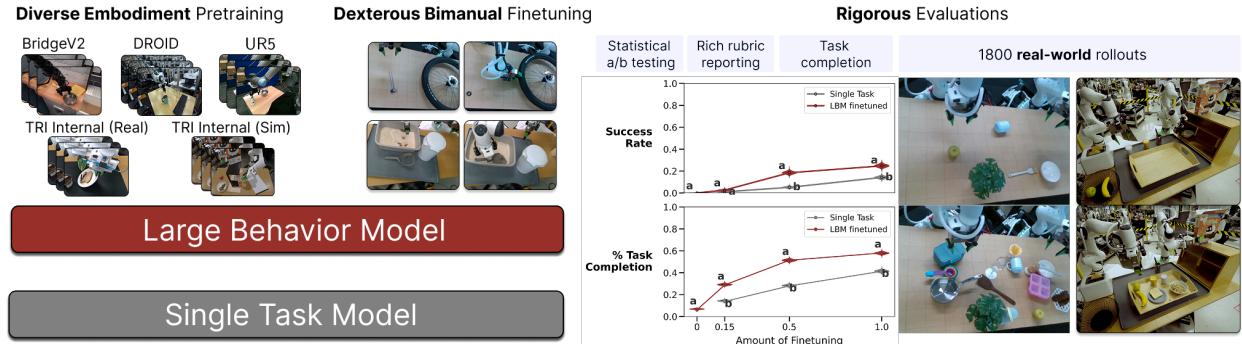


Figure 1: Large Behavior Models (LBMs) are visuomotor policies trained on a diverse corpus of simulation and real-world manipulation data. Through a carefully designed evaluation pipeline that leverages both simulation and real-world experiments, we show that finetuned LBMs are more robust to distribution shift and achieve better performance than single-task baselines. Moreover, when finetuning on novel tasks, LBMs require a fraction of the data to achieve the same performance as baseline methods.

II. RELATED WORK

A. Robot Learning at Scale

Robot learning is undergoing a paradigm shift towards creating generalist manipulation policies [2], [4]–[9], inspired by the scaling hypothesis successfully applied in Natural Language Processing [14], [15] and Computer Vision [10]–[13]. This shift is driven by the creation of large-scale and diverse datasets [16]–[18] as well as high-capacity models, particularly transformer-based Vision-Language-Action (VLA) models [4], [19]–[25], which have become central, integrating perception, language, and action within a unified framework trained largely via imitation learning. A key driver of VLA performance is transferring knowledge from large pretrained foundation models [26] bringing semantic understanding, improved reasoning, and strong visual representations, which when finetuned on robotics datasets, enable capabilities like zero-shot task execution. The choice of action representation—whether discretized into tokens [27]–[29], directly regressed as continuous commands [2], or generated through diffusion models [4], [5]—affects the policy’s ability to produce precise, multimodal, and real-time behaviors. Despite progress in training generalist policies, challenges such as catastrophic forgetting, data heterogeneity, scarcity of high-quality data, multimodal fusion, handling dexterity, and maintaining real-time inference speed remain open research problems. This work focuses on rigorously evaluating the effects of multi-task pretraining (as opposed to, for example, architectural novelty), and studies a fixed policy architecture (described in Section IV-B2) throughout.

B. Datasets for Robot Learning

Training generalist robot policies requires large-scale, diverse datasets, yet acquiring this data poses signifi-

cant challenges. Unlike large-scale language and vision datasets [11], [15], [30]–[35], which are generally derived from internet sources, collecting robotics data in the real world is inherently slow and expensive. Robot data is most commonly collected via teleoperation, wherein human operators remotely control robots, yielding high-fidelity, embodiment-specific data. Large datasets such as RT-1 [28], Bridge [36], RH20T [37], DROID [16], and AgiBot [38], among others, have been collected using this approach, often over the course of months or years with multiple robots. Pooled datasets like Open X-Embodiment [17] aggregate teleoperation and other types of robot data from numerous labs (over 1 million trajectories and 22 embodiments) with the hope of training policies that benefit from transfer across robot embodiments. Simulation [39]–[41] provides one promising alternative for cost-effectively generating robot manipulation data at scale [42]–[45]; however, the differences between simulators and the real world present challenges. One method of overcoming these differences is to simultaneously train (“co-train”) on data from both domains [46], [47]. We use sim and real co-training in this work in order to more effectively evaluate LBMs that were trained primarily on real-world data in simulation. Another approach to quickly and cost-effectively collect data is to circumvent the need for robots entirely through the use of specialized devices manually controlled by people [48]–[51]. In this work, we train LBMs on a mixture of data (described in Section IV-D) sourced from openly available datasets as well as from our internal data collection efforts in the real world (using both teleoperated robots and specialized devices [48]) and in simulation with the goal of better understanding the value of training on these large-scale, multi-task datasets.

C. Evaluating Robotic Manipulation Policies

Measuring the performance of LBMs, either for research or real-world deployment purposes, requires reproducible, reliable, and scalable evaluation methods and frameworks [52]. Absence of standardized hardware makes consistent benchmarking a challenge. As a result, most benchmarks are simulation-based with notable examples including RLBench [45], ManiSkill [53], Meta-World [54], Robosuite [55], BEHAVIOR [56], and RoboTHOR [57]. Evaluation within these benchmarks typically relies on quantitative metrics such as success rate, task completion percentage, and completion time, and emphasizes generalization (for example, to unseen objects, tasks, or scenes) or sample efficiency. While prior work in navigation highlights simulation-to-reality gaps caused by dynamics and visual discrepancies [57]–[59], evaluation of manipulation policies poses additional challenges due to tighter robot and environment coupling and the sensitivity of task outcomes to subtle variations. SIMPLER [60], a recent simulation framework, mitigates control and visual discrepancies between real and simulated environments through the use of system identification [61], [62] and various image editing and matching techniques. Alternative real-world evaluation approaches focus on establishing standardized object sets, tasks, datasets, and evaluation protocols [63]–[70], remote access to shared robots [71]–[73], or improving evaluation efficiency [74], [75]. Despite progress, challenges remain in evaluating generalist robotic manipulation policies across many diverse tasks, reliably benchmarking complex long-horizon interactions, and assessing robustness and safety critical aspects in dynamic environments.

III. RESULTS

We aim to achieve a nuanced understanding of LBM performance under a number of real-world conditions. Our main hypotheses are that due to pretraining, 1) new tasks can be learned with less data; 2) policies achieve better performance; and 3) policies are more robust under distribution shift. For each individual task, we compare against a single-task policy trained from scratch. We use simulation and real-world (hardware) experiments to measure LBM performance on tasks for which demonstrations were seen during pretraining (“seen” in the following) as well as on novel tasks that were not used for pretraining (“unseen” in the following). We evaluate the policies under two conditions, nominal and distribution shift; we systematically create distribution shifts as described in Sec. IV-E. Task complexity varies from simple pick-and-place manipulation to long-horizon tasks requiring a high degree of dexterity, such as coring an apple, setting a breakfast tray, or installing a bike rotor. Each real-world task was evaluated with 50 rollouts per task per policy per

condition. Simulation tasks were run 200 times per task per policy per condition; due to missing data, a few tasks were analyzed with fewer than 200 rollouts, see Section VIII-D for details.

Section IV-A describes our evaluation protocol and our process for creating controlled and repeatable distribution shifts. Our metrics for measuring performance are success rate (SR) and task completion (TC). Success rate, while a useful and important signal that is standard across robot learning publications, does not tell the full story of policy performance [52]. There is a notable difference between a policy that almost, but not quite, succeeds and one that does nothing. To capture and quantify this nuance, we create rubrics for real-world evaluation and predicates for simulation evaluation, as described in Section IV-A2. Using these, we measure task completion, based on task-specific intermediate milestones. For real-world evaluation, task completion is evaluated by filling out rubrics manually; we created a quality assurance (QA) process to measure the reliability of the rubrics. For simulation, task completion is computed automatically.

Although we report absolute success rates, the most important results are the *relative* success rates of the different methods. The absolute success rates are very task dependent and can easily be shifted up or down based on how difficult we make the task (e.g., by broadening the distribution of initial conditions) and/or by changing the number of task-specific demonstrations. We design our experiments to make tasks quite difficult—targeting policy success rates around 50%—so that the relative success rates are as informative as possible, but in practice end up with significantly higher or lower rates.

In the following, as described in Section IV-A4, we use violin plots to convey the quantitative results with a horizontal line indicating the mean. For SR, the mean is the empirical success probability (successful runs/all runs) and the violin is the Bayesian posterior of individual success rates under a uniform Beta prior; it represents the uncertainty in the true success probability given the success rate and total number of runs. For TC, the mean is the mean of the task completion across all runs in either individual tasks or the aggregate of all tasks. For individual tasks, the violin represents the full data distribution of TC for that task. In plots where the tasks are aggregated, the violin represents the Bayesian posterior of the mean TC under a uniform Dirichlet prior. We emphasize that these distributions are evaluated for each individual policy checkpoint, and do not capture the randomness from the training process.

For each task we perform multi-task hypothesis checking to test whether the separation observed is statistically significant. We indicate statistical significance in the result plots using the Compact Letter Display (CLD) method, which assigns the same letter to policies that cannot be

separated and different letters to indicate separation with statistical significance.

In the following we discuss “seen” tasks (Section III-A), “unseen” tasks (Section III-B), and the effects of the amount of data used to pretrain and finetune LBMs on the performance (Section III-C).

A. LBM performance on “seen” tasks

We first analyze LBM performance on a subset of tasks included in the pretraining dataset (i.e., “seen” tasks), with results summarized in Figure 2. We evaluate LBMs that are only pretrained (Figure 2, teal) as well as pretrained LBMs with additional finetuning on individual tasks (Figure 2, maroon). Since these are relatively simple, short-horizon tasks, we do not present task completion results. The top row represents nominal conditions, and the bottom row represents experiments conducted under distribution shift. Details regarding how we systematically created the distribution shift are in Section IV-E1 for simulation and in Section IV-E2 for the real world.

For “seen” tasks, we expect the pretrained LBM’s success rate to be greater than zero since the tasks were in the training data, and we expect the finetuned LBMs to perform better than the single-task baseline, given the information that is encoded as part of the pretraining. In this set of experiments, we find that:

Finetuned LBMs perform better on “seen” tasks than the single-task baselines: From Figure 2, we see that when aggregating over tasks, the finetuned LBM performs better than the single-task baseline under nominal and distribution-shift conditions both in simulation and in the real world; the finetuned LBM is statistically distinguishable from single-task in all the aggregate plots.

When looking at individual tasks, the finetuned LBM is statistically the same or better than the single-task policy (i.e., it is labeled with “a”) in 3/3 real-world tasks and 15/16 sim tasks, both for the nominal conditions and the distribution shift. Interestingly, the one sim task in which single-task statistically outperforms the finetuned LBM is different between the experimental conditions, as further discussed below. Overall, the finetuned LBM is statistically better than single-task policies under nominal conditions in 2/3 real-world tasks and 3/16 simulation tasks; under distribution shift, finetuned LBM statistically outperforms single-task policies in 2/5 real-world tasks and 10/16 simulation tasks.

Finetuned LBMs are more robust to distribution shift on “seen” tasks than the single-task baseline: As expected, and as seen when comparing the two rows of Figure 2, when we introduce distribution shift the overall task performance, here defined as success rate, drops on most tasks. However, we also observe that the finetuned LBMs go from statistically outperforming single-task policies on 3/16 policies in simulation under nominal

conditions to 10/16 under distribution shift. Furthermore, for the aggregate simulation plots, the separation between finetuned LBM and single-task widens under distribution shift. These results suggest that finetuned LBM created policies that are more robust to distribution shift than policies trained from scratch; this result also holds for the “unseen” tasks, as described in Sec. III-B.

LBMs without finetuning have nonzero success rate on “seen” tasks and exhibit similar performance to the single-task baselines: As expected, the pretrained LBM (without any task specific finetuning) has a success rate that is greater than zero on all tasks under nominal conditions. When aggregating over tasks, pretrained LBM is statistically indistinguishable from single-task in simulation (both conditions) and in real-world with object-centric distribution shift. On nominal real-world and station distribution shift, pretrained LBM performs worse than the single-task baseline. For individual tasks, in all real-world tasks, pretrained LBM is either statistically indistinguishable (5/8) or worse than (3/8) the single-task baseline, across both conditions. In simulation, under nominal conditions, pretrained LBM is statistically better than single-task in 3/16 tasks, and is statistically worse than single-task in 4/16. When considering distribution shift, the situation is similar, with pretrained LBM statistically better than single-task in 4/16 tasks and worse in 2/16 tasks. We make two observations related to the pretrained LBM; first, it performs worse than single-task mainly in the real-world tasks. Second, as discussed in Section IV-D2, we discovered an error in the pretraining process (data normalization) after the evaluations were complete. This error might also affect the pretrained LBM’s performance (see Section XIV).

We further investigate specific task performances that we found surprising.

There are cases where the finetuned LBM performs statistically worse than the single-task baseline: In each condition, there is one simulation task for which finetuned LBM statistically underperforms the single-task baseline; the under-performing task depends on whether evaluation was conducted under nominal conditions or distribution shift.

Under nominal conditions, *TurnCupUpsideDown* finetuned LBM performs substantially worse than both pretrained LBM and the single-task baseline, with a success rate of 0.335. Here, in almost half (89/200) of the rollouts for finetuned LBM on this task, the robot did not move away from its initial pose before the simulation times out, as shown in Fig. S13(e-left). When introducing distribution shift, we did not observe this behavior (Fig S13(f-left)). Under distribution shift, *StackPlatesOnTableFromRack* performs worse than the single-task baseline. Inspecting the policy behavior did not reveal a systematic qualitative failure mode as in the

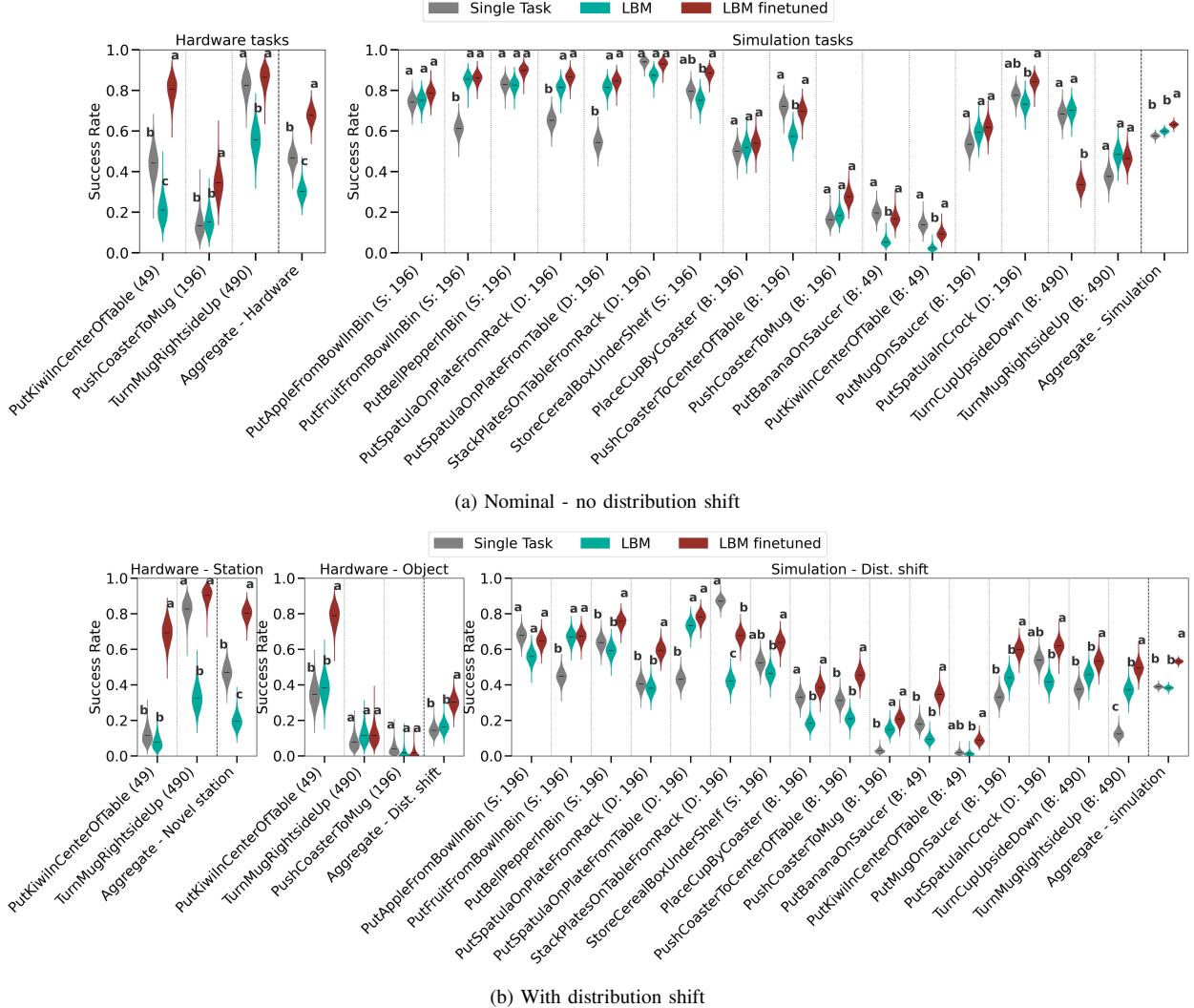


Figure 2: LBM performance on “seen” tasks in real-world and in simulation without (a) and with (b) distribution shift. We compare single-task with pretrained LBMs and with LBMs after finetuning. The x-axis labels show the task name, scenario name (for simulation tasks), and the number of demonstrations. Violin plots represent Bayesian posteriors of success rates under a uniform Beta prior and the observed success/failure data; policies labeled with different letters are statistically distinguishable.

previous case but confirmed the poor policy performance in general.

Performance on the *Breakfast* scenario (simulation tasks): Our simulation tasks are grouped into a handful of “scenarios”, which start with the same environment assets and contain thematically similar tasks (see Section IV-C2 for more details). We see that tasks belonging to the *Breakfast* scenario (denoted by *B*: in the X-axis caption of Fig 2, right) have higher variance in task success across tasks, compared to other scenarios, and that three tasks have substantially lower performance than the rest. We attribute this to two factors: first, for *PutBananaOnSaucer* and *PutKiwiInCenterOfTable* we have only 49 demonstrations; as expected and as discussed in the following

sections, the less demonstration data available, the worse the performance. Second, for *PushCoasterToMug*, we observe that the task is more difficult than the rest because the robot needs to push obstacles out of the way, the goal (mug) can be anywhere on the table, and all the demonstrations use a pushing strategy on the side of coaster, which is sensitive to variance in the height of the end effector. We provide additional details in Section X.

A few tasks were more successful in the real world than in simulation: For the “seen” tasks, all the tasks that were evaluated in the real world were also evaluated in simulation. Given the lack of uncertainty in simulation, we would expect under nominal conditions that the success rate of these tasks would be higher in simulation. This is

not the case for two tasks: *PutKiwiInCenterOfTable* (*Kiwi* for short) and *TurnMugRightsideUp* (*Mug* for short); for these tasks, across all policies, the real-world success rate is higher than simulation. One hypothesis for *Mug* is that the simulation timeout (the maximum time a simulation rollout is run for) is too short for the policy to eventually succeed. In contrast to the real-world evaluation, where the operator decided when to terminate the rollout, in simulation the rollout will be stopped even if the task is about to be completed. Computing success rates on real-world rollouts by truncating to the simulation timeout yields a similarly low success rate. For *Kiwi* we observe different behavior between simulation and hardware and will continue to explore this discrepancy. We present further analysis in Section X-C. Simulation and real-world evaluations under distribution shifts are not directly comparable due to the differences in how they are introduced respectively.

B. LBM performance on “unseen” tasks

We explore LBM performance on tasks that do not appear in the pretraining dataset, i.e., unseen during training. It is straightforward to make “unseen” tasks in simulation. First, we generate additional tasks in the same scenarios as the “seen” tasks and that are similar in complexity, Figure 3. For the real world, generating tasks that are meaningfully distinct from our multi-year data collection effort is more challenging. To address this, we designed several long-horizon, multistep, dexterous tasks that were outside of the scope of previous task collections. For parity, we also added a new simulation scenario, *Kitchen*, with similarly long-horizon, multistep, dexterous tasks. These complex tasks are designed to test the limits of what our policies are capable of executing. Our results for the complex tasks are summarized in Fig 4 for nominal conditions and in Fig 6 for evaluation with distribution shift (sim only). In both of these figures, we present success rate results on the top row and task completion results on the bottom row. Furthermore, in addition to the SR and TC results for policies created (trained or finetuned) with all the task data, we analyze the aggregate performance of the policies when finetuned (LBM) or trained (single-task baseline) with subsets of the data (plots on the right). We provide a similar analysis for one real-world task in Figure 5.

For our “unseen” tasks, especially the complex tasks, we do not expect the pretrained LBM to succeed; we therefore only compare the finetuned LBM and the single-task baseline. Furthermore, for the complex tasks, we expect low success rates and more intuition gained from the task completion plots.

Finetuned LBMs perform better on “unseen” tasks than the single-task baseline, both in nominal conditions and under distribution shift: When aggregated across tasks, the finetuned LBM is statistically better than

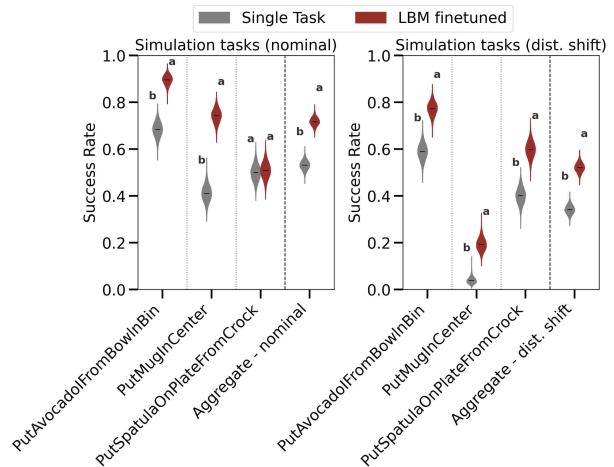


Figure 3: **LBM performance on “unseen” simulation tasks from scenarios that are part of the simulation training set.** **Left:** evaluation done under nominal conditions. **Right:** evaluation done under distribution shift. Violin plots represent Bayesian posteriors of success rates under a uniform Beta prior and the observed success/failure data; policies labeled with different letters are statistically distinguishable.

the single-task baseline in the real world and in simulation (both nominal and distribution shift), and across metrics (success rate and task completion).

For the simpler simulation tasks, finetuned LBM is statistically better than single-task on 2/3 tasks in nominal conditions and all the tasks under distribution shift. For the complex tasks, while as expected the success rate is low, and even lower once considering distribution shift, some tasks still show statistical separation, and in all of those the finetuned LBM is more successful (2/5 tasks for real-world, 3/5 tasks in nominal simulation, and 1/5 tasks in simulation with distribution shift).

When considering task completion, the conclusion that finetuned LBMs outperform single-task baselines becomes clearer; finetuned LBM is statistically better than the single-task baseline in 4/5 real-world tasks, and in 4/5 simulation tasks both in nominal conditions and under distribution shift. Visually inspecting the data distribution for task completion indicates that the finetuned LBM is able to achieve more steps of the task compared to the single-task baseline. When considering real-world tasks, we see that there is a task, *SetBreakfastTable*, that the single-task policy never completes,¹ while the finetuned LBM succeeds and achieves higher task completion; conversely, there are two tasks, *BikeRotorInstall* and *CutAppleInSlices*, where in all rollouts the finetuned LBM

¹Due to the Bayesian analysis prior assumptions, the success rate graph appears to have a mean that is greater than 0; the empirical success rate for the single-task baseline of *SetBreakfastTable* is in fact 0.

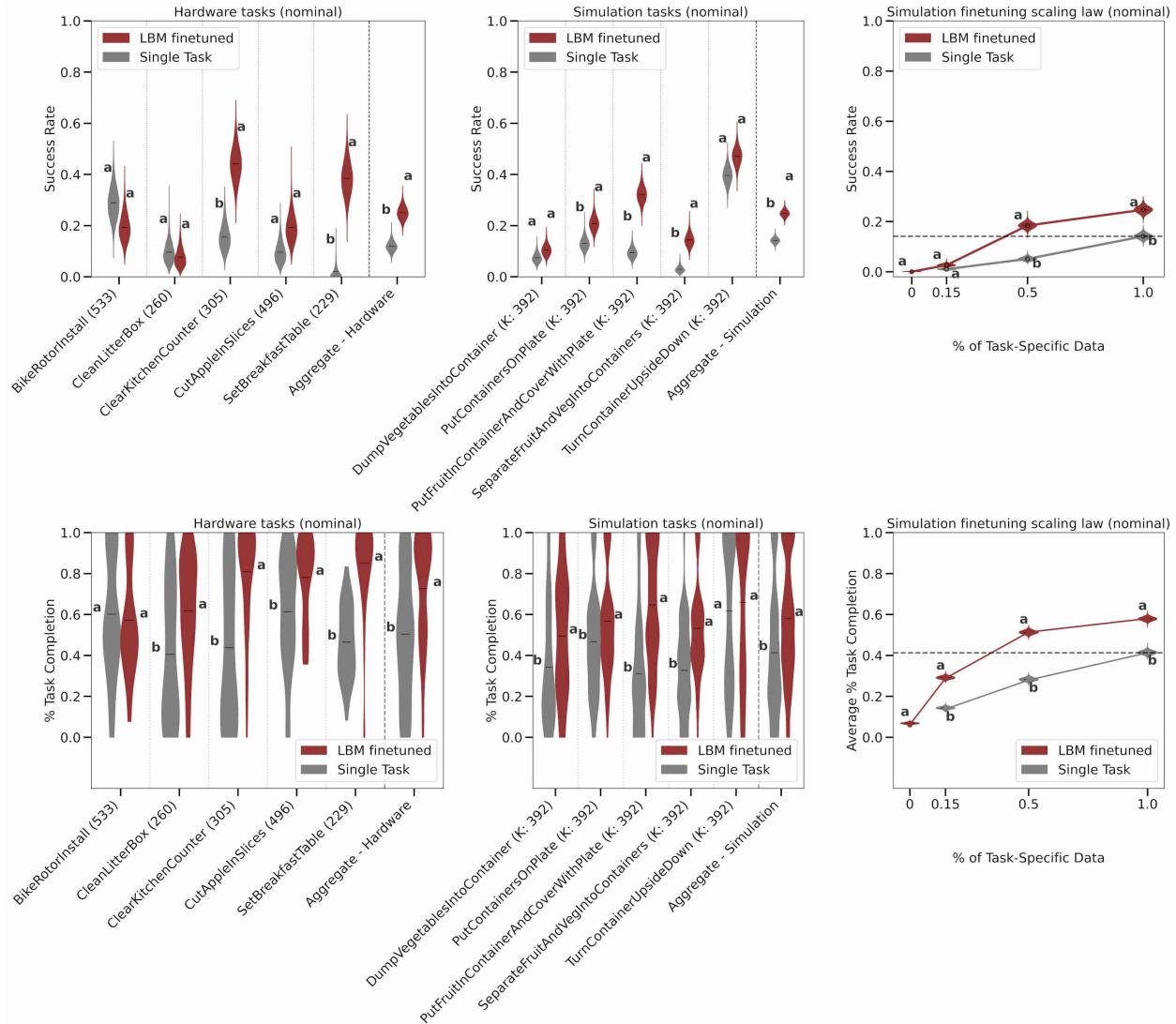


Figure 4: LBM performance on “unseen” tasks in real-world and in simulation evaluated under nominal conditions. We compare the single-task baseline with LBMs after finetuning. The top row shows success rate results while the bottom row shows task completion results. The x-axis labels show the task name, scenario name (for simulation tasks), and the number of demonstrations. Violin plots for SR represent Bayesian posteriors of success rates under a uniform Beta prior and the observed success/failure data. For TC, violin plots of individual tasks (left) represent the entire data distribution; we use statistical hypothesis tests over the mean TC for the CLD letters shown for these plots, and we provide the Bayesian posteriors used for the tests in Fig. S17. The violin plots for TC as a function of percentage of data (right), the plots represent the Bayesian posterior of the mean TC under a uniform Dirichlet prior. Policies labeled with different letters are statistically distinguishable.

completed part of the task, while the single-task baseline sometimes completely failed.

Finetuned LBMs require less task-specific data to achieve similar performance as the single-task baseline: In Figures 4 and 6, the rightmost column shows the finetuned LBM and single-task baseline performances when aggregating across all five simulation tasks; each data point corresponds to finetuning/training with a different fraction (by demonstration) of the available task-specific data. For task completion, across both conditions

(nominal and distribution shift), for all fractions of data, finetuned LBM is statistically better than the single-task baseline. For success rate, since the overall success rate is low, especially under distribution shift, the finetuned LBM is statistically better starting at 50% of the data. In aggregate, and interpolating, we see that to achieve similar performance in simulation, when finetuning an LBM we require less than 30% of the data needed for training from scratch.

We performed a similar experiment on the *SetBreak-*

fastTable real-world task, as shown in Figure 5 where the violin plot represents the full data distribution. LBM finetuned with only 15% of the data, statistically outperforms the single-task baseline (trained on all the data), further supporting our simulation findings.

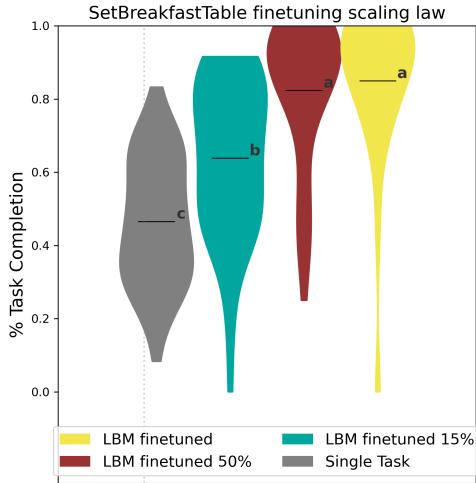


Figure 5: Performance of LBMs finetuned on a fraction of *SetBreakfastTable* data compared with the single-task baseline. We highlight that the LBM finetuned with 15% of data is already outperforming the baseline trained with all the data.

C. Pretraining scaling laws

Using simulation, we explore the scaling laws of LBM pretraining—the effects of pretraining dataset size on the performance of LBMs, measured through task completion on “unseen” tasks. Due to the complexity of the tasks, the success rates are low (as seen in Figure 4) and do not provide a statistically distinguishable conclusion, therefore we present only task completion results here.

We evaluate on the same five “unseen” simulation tasks as in Sec. III-B and perform experiments under nominal conditions, where we compare LBMs that were pretrained with different fractions of the data (Section IV-D) used to train the LBM of the previous subsections. We create four pretraining datasets for this experiment: 1) the full dataset (**OXE-Ramen** and **TRI-Ramen**), referred to as **LBM finetuned**, 2) **TRI-Ramen** only, referred to as **LBM finetuned [TRI-Ramen]**, 3) 50% of all tasks present in the **TRI-Ramen** dataset, referred to as **LBM finetuned [TRI-Ramen-50%]**, and 4) 25% of all tasks present in the **TRI-Ramen** dataset, referred to as **LBM finetuned [TRI-Ramen-25%]**. The results are shown in Figure 7; each line represents pretraining with a different fraction of data then finetuned, or the single-task baseline. For single-task, the graph starts from 15% of the finetuning data because without data (0%) we do not have a single-task

policy. Note that the performance of the single-task and finetuned LBM are consistent with the previously reported results in Sec. III-B and specifically in Fig. 4. Similarly to Figures 4 and 6, each point on the X-axis indicates the percentage of total available demonstration data used to finetune the LBM.

We note consistent separation with statistical significance between LBM finetuned and the TRI-Ramen version when using 0%, 15% and 50% finetuning data. Interestingly, when using 15% finetuning data, all five models are separable with statistical significance, with performance steadily increasing as we add more pretraining tasks. The same trend holds when using 50% and 100% pretraining data, and all models achieve the best performance when pretraining and finetuning with all available data.

Pretraining and finetuning data tradeoff: This experiment suggests that there is a tradeoff between pretraining and task-specific finetuning. If there is limited task-specific data for finetuning, more tasks/data in the pretraining dataset corresponds to better finetuned LBM performance, even if the data is diverse (here the OXE dataset). Conversely, if there is a lot of task-specific data, LBMs pretrained with less data might suffice.

IV. MATERIALS AND METHODS

In this section we describe our evaluation protocol, the architecture we use to train the models, details regarding the experiments, the tasks, and the data we use to train and evaluate LBMs.

A. Evaluation Protocol and Analysis

One of the main contributions of this paper is our focus on rigorous robot policy evaluation that goes beyond what is typically done in the robot learning community. We designed an evaluation protocol that aims to ensure repeatable yet diverse initial conditions, and fairness across policy candidates. We evaluate our hypotheses in both simulation and the real world using statistical tools; this section describes the protocol we used, and the statistical methods we employ to test our hypotheses.

1) Policy comparison protocol: We focus on fair comparison across the different policies (single-task models vs LBMs), that is, we make every effort to minimize bias and subject the different policies to similar testing conditions, including environmental and initial conditions. To do so, in all our evaluations, the evaluators did not know which policy was being tested (blind testing), policy ordering was randomized to maximize fairness for each policy, and the initial conditions were consistent across the policies, within human error for hardware [52].

In simulation, we use the same simulation parameters and initial conditions (set via random seed) for all policies being compared. Initial conditions are sampled from the

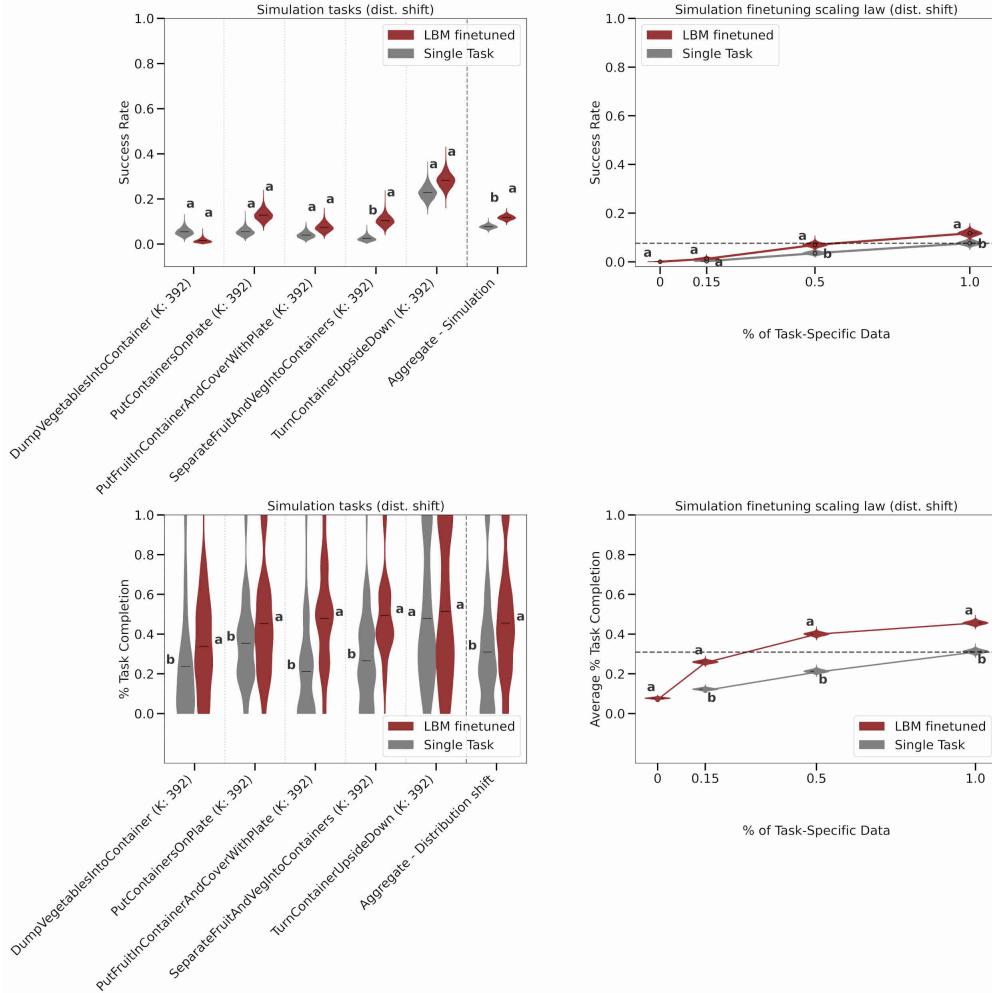


Figure 6: LBM performance on *unseen* tasks in simulation evaluated under distribution shift. We compare the single-task baseline, with LBMs after finetuning. The top row shows success rate results while the bottom row shows task completion results. The x-axis labels show the task name, scenario name (for simulation tasks), and the number of demonstrations. Violin plots for SR represent Bayesian posteriors of success rates under a uniform Beta prior and the observed success/failure data. For TC, violin plots of individual tasks (left) represent the entire data distribution; we use statistical hypothesis tests over the mean TC for the CLD letters shown for these plots, and we provide the Bayesian posteriors used for the tests in Fig. S18. The violin plots for TC as a function of percentage of data (right), the plots represent the Bayesian posterior of the mean TC under a uniform Dirichlet prior. Policies labeled with different letters are statistically distinguishable.

same distribution from which associated training data was generated, unless we are explicitly testing out-of-distribution generalization, but these initial conditions are new samples from the training distribution—we do not reuse exact initial conditions between training and evaluation in simulation.

To ensure fairness when evaluating policies on hardware, we split evaluation into test bundles with the size of each bundle equal to the number of policies being compared. Each bundle corresponds to a single initial condition with a randomized policy ordering. After evaluating each policy, the initial conditions are reset until the bundle is completed and a new initial condition

is evaluated. Real-world evaluation is blind—we ensured that the evaluator had no knowledge of which policy was being evaluated during each run. As for ensuring consistent test conditions, running policies in bundles with randomized order mitigates the effects on policy performance from environment changes (e.g., lighting). For initial conditions, we created a workflow where the robot evaluator was given an image overlay of the desired visual scene (see Fig. 8); by matching the robot’s environment to the overlay, we mitigate the uncertainty in the initial conditions. The initial conditions come from simulation if the task and conditions are modeled in simulation, or from real pictures of initial robot scenes

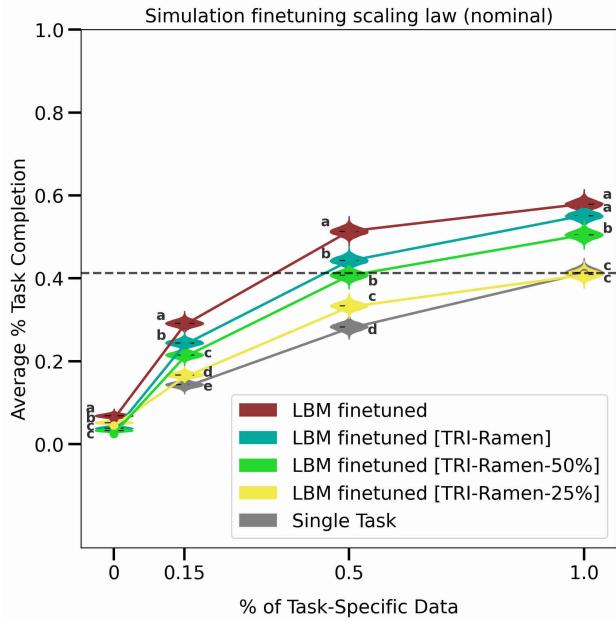


Figure 7: LBM performance when pretraining with varying subsets of the full pretraining dataset. We evaluate under nominal conditions on five “unseen” tasks in simulation and report the mean of the task completion metric across all tasks. Violin plots represent Bayesian posteriors of the task completion mean under a uniform Dirichlet prior and the observed task completion scores; policies labeled with different letters are statistically distinguishable. Each data point corresponds to 200 rollouts per task, for a total of 1000 rollouts.

in the case of real-world tasks. We provide additional details for setting up initial conditions in simulation in Section VIII-B and for real-world in Section IX-C.

2) Rubrics and predicates: Rubrics are a set of questions that the robot evaluator fills out as they are observing the robot behavior. The questions address task progress (e.g., “robot grasped the apple”) and failures (e.g., “robot dropped apple”). The questions are binary yes/no questions that can then be statistically analyzed to enrich our understanding of policy performance. The rubrics include a set of milestone questions per task, where each milestone is a step necessary towards a successful rollout. We calculated a Task Completion rate for the real-world rollouts by assigning a +1 credit to a successfully completed milestone and 0 otherwise, then the credits are summed and divided by the number of milestones.

Policy evaluation in simulation is automated through predicates that are defined over the simulation state (e.g., a predicate that is True when the apple center is closer than a small threshold to the bin center). A success in simulation corresponds to a set of predicates being True. These predicates also allow us to analyze partial success and failure modes of the policy.

For more details and example rubrics and predicates,

refer to Sections IX-D and VIII-C.

3) Rubric QA: Considering that the rubric results are provided by reviewers who are prone to human error, we estimated the discrepancy percentage through an additional validation on a smaller subset of the reported rubric answers. To this end, we conducted a quality assurance (QA) round on $\sim 27\%$ of nearly 2700 real-world evaluation rollouts² to estimate the discrepancy in the answers. The subset of people doing rubric QA was separate from the robot evaluators. The QA success rate discrepancy was 2.31% and the overall rubric question discrepancy was 6.25%.

We performed one round of corrections, based on the discrepancy between the recorded success and the success calculated based on the rubrics. This involved updating five rollouts. We note that the results reported in Section III were calculated after the corrections were applied.

4) Statistical Analysis: Performance Characterization of Individual Policies: Throughout the paper, we report statistical uncertainty of empirical performance of individual policies. Our analysis is based on binary success/failure and on task completion for more challenging, longer-horizon tasks.

a) Binary Success/Failure Criteria: From the statistical perspective, computing the success rate of a policy is equivalent to estimating the Bernoulli parameter p of the underlying Bernoulli distribution generating the success/failure labels. This assumes that each evaluation trial is independent and identically distributed (i.i.d.). In simulation, this assumption is satisfied by randomizing the initial conditions with seeds. For real-world experiments, we take a set of measures discussed in Section IV-A1 to mitigate the effect of time-varying randomness and unwanted bias as much as possible.

In some cases, we report statistical results that are aggregated over multiple tasks. Strictly speaking, this violates the i.i.d. assumption as the number of trials from each task is fixed *a priori* instead of a random draw from a uniform distribution over the 3 tasks. Nevertheless, the effect of this violation is negligible in practice and is recommended practice in prior work on policy evaluation [76]. To visualize the uncertainty over the unknown parameter p , we perform Bayesian analysis and compute the posterior after observing success/failure data. Specifically, we use a uniform prior over the Bernoulli parameter as suggested by [52] and plot the Bayesian posterior in the form of a violin plot. See Fig. 2 for an example. We use violin plots rather than standard Confidence Intervals (CIs) for two reasons. First, violin plots depict the entire distribution of the parameter p instead of a single interval, thus presenting richer information on statistical uncertainty.

²Not all real-world rollouts are included in the results, since some came from earlier iterations. All rollouts that were QAed were from evaluation tasks.



Figure 8: **Initial conditions overlay.** **Left:** overlay of current camera image and desired initial condition for the experiment. **Middle:** desired initial condition, created using simulation or from pictures of initial robot scenes; the operator will try to recreate this by manipulating the scene elements present. **Right:** current camera view.

Second, CIs can be confusing or misleading when it comes to policy comparison; one may wrongly conclude that two results are not separated with statistical significance if their corresponding CIs overlap [77]. In fact, CIs can overlap while more powerful hypothesis tests may still statistically separate them. In Section IV-A5, we discuss how we can incorporate such hypothesis tests in our analysis to accompany the individual Bayesian analysis.

b) Task Completion Criteria: For the task completion criteria based on our rubrics or predicates, the mean of the corresponding categorical distribution may not represent the entire distribution, unlike the Bernoulli case (i.e., variance and other higher-order moments are not uniquely determined by the mean). Therefore, we present the violin plots of the raw data instead of the Bayesian analysis of the mean so they illustrate the full distribution better; see Fig. 4 (bottom row, left and middle) for an example. We will resort to the mean of the distribution only to a) examine the effect of fractional fine-tuning (Fig. 4, bottom row, right) and b) compare the performance of two or more policies. For a), we use a uniform Dirichlet prior, similar to the binary success/failure setting. For b), we discuss the details of policy comparison below.

5) Statistical Analysis: Performance Comparison of Multiple Policies: We perform hypothesis tests to compare multiple policies; we run $k(k - 1)/2$ pairwise tests where k is the number of policy models to be compared at once. In each test, the confidence level is adjusted for multiplicity via Bonferroni correction unless otherwise noted, so that a global 95% confidence level is maintained across all the $k(k - 1)/2$ comparisons. We use the Compact Letter Display (CLD) algorithm [78] to summarize the results. With CLD, each policy is labeled with one or more letters such as “a”, “ab” or “bc”. Two policies that do not share the same letter are separated with 95% confidence.

For each pairwise test, we use an appropriate statistical method depending on the type of the data. For the more common binary success/failure criteria, we use a sequential hypothesis testing framework as suggested in [75], which sequentially compares paired outcomes

of each evaluation trial until either a decision is reached or all the trials are consumed. Specifically, we adopt the test originally proposed in [79], which yields reasonable statistical power in the small sample size regime while maintaining a strict Type-I error control for binary data. (That is, the chance of falsely concluding that the two policies are statistically separated is upper-bounded by 5%). For the task progress, we cannot apply [79] as it does not readily extend to categorical data. Instead, we use the Welch’s t-test [80] to compare the means of two distributions. Strictly speaking, the discrete nature of the data violates the underlying assumption of the t-test that the data is normally distributed. Thus, the Type-I error is not controlled. However, we ensure that the sample size is sufficiently large (i.e., about 50 or more) so that the degree of violation is small owing to the central limit theorem.

Nominally, the aforementioned policy comparison procedure is performed per task. When multiple tasks are considered and plotted at once, we do not further adjust the confidence level of individual pairwise tests, since doing so would yield individual confidence levels that are too stringent. Therefore, we note that the Type-I error is not globally controlled across tasks when they are presented in one plot. Nevertheless, we do aggregate the results over tasks when we compare the overall multi-task performance of different policies. This yields an exchangeable Bernoulli sequence, approximating an i.i.d. sequence by marginalizing over skills. Exchangeability is weaker than i.i.d., so the underlying assumption of the pairwise test is slightly violated. However, we empirically verified that this difference does not affect the statistical validity of results.

B. Large Behavior Models

In this section we describe the LBM generative model, training objective, architecture, pretraining and finetuning recipes, and deployment details; see Figure 9 for an overview of our architecture.

1) Diffusion for Visuomotor Control: We implement generative policies for visuomotor control by employing

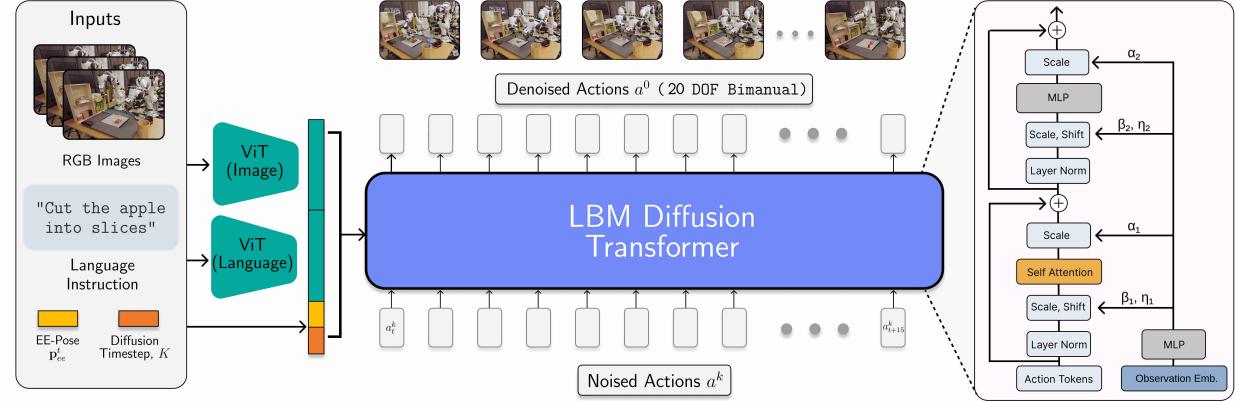


Figure 9: LBM Architecture: we use a Diffusion Transformer [81] conditioned on language, vision and proprioception and which outputs 20-dimensional actions over 16 timesteps (see Section IV-B2 for details). During deployment, we run the policy at 10 Hz and the robot executes the predictions over the first 8 future timesteps before replanning actions.

Denoising Diffusion Implicit Models (DDIM) [82]. We choose this class of generative model because it has been shown to be effective at learning visuomotor manipulation policies from human demonstrations [1]. DDIMs transform a simple prior distribution, typically Gaussian noise, into a complex, structured (action) distribution conditioned on input data – in our case, visual, proprioceptive, and language observations. This transformation uses a deterministic sampling process derived from denoising diffusion probabilistic models (DDPM) [83]. Given $K \geq 1$ denoising steps, we start with a noise sample $A_t^K \sim \mathcal{N}(0, I)$ at time t and use DDIM to denoise it into a continuous action A_t^0 in K iterative steps. In order to predict actions conditioned on observation inputs, we modify the original DDIM update as follows:

$$A_t^{k-1} = \alpha \left(A_t^k - \gamma \cdot \epsilon_\theta(O_t, A_t^k, k) \right) \quad (1)$$

where A_t^k is a set of noisy actions at the k -th denoising step, O_t are the observations, and k is the diffusion timestep. Parameters α and γ are determined by a noise schedule which varies with the diffusion timestep k , and ϵ_θ is the noise-prediction neural network with weights θ . To train ϵ_θ , we sample an action A_t^0 and a random step k , and add a step-dependent Gaussian noise ϵ_k to form a noisy action $A_t^k = A_t^0 + \epsilon_k$. The network is then trained to predict ϵ_k from A_t^k , enabling it to denoise across the full range of diffusion levels. Mathematically, we optimize the following DDPM loss with respect to θ :

$$\mathcal{L}(\theta) = \| \epsilon_k - \epsilon_\theta(O_t, A_t^k, k) \|_2^2. \quad (2)$$

2) *Policy Architecture:* We parametrize the noise prediction network, ϵ_θ , as a Diffusion Transformer (DiT) [81], which conditions on features extracted from the observations and the diffusion timestep in predicting actions (see Section IV-D1 for the observation and action spaces).

To extract features from the image observations we use the CLS token output from a pretrained CLIP Vision Transformer (ViT) backbone [11]. Language features are similarly computed from the task description using a CLIP text encoder, with a projection layer on top of the pooled End of Sequence token. The language and visual features are concatenated with the proprioception for each observation timestep as well as with the diffusion timestep, k , which is encoded with a sinusoidal positional embedding [83] followed by a two-layer MLP. During training, we finetune through the visual feature extractor, which is shared across all camera inputs. We keep the language-feature extractor frozen, but train a projection layer on top of the language features.

The DiT conditions on two timesteps of concatenated observation features, which together have a size of 6,732, and the encoded diffusion timestep via an adaptive layer norm (adaLN) MLP [81]. This model consists of eight DiT blocks with an embedding size of 768. The network predicts 16 timesteps of 20-dimensional actions, for a total output size of $A_t = 320$. All experiments use the architecture described above. In the case of finetuning on single-task data or evaluating from-scratch policies we use the same architecture and only use the language prompts for the task of choice.

3) *Training and Deployment:* Our training recipe follows a common pattern for foundation models where we first pretrain policies on the full data mixture and then finetune on narrower data subsets [14], [15], [29], [35], [84]. Hyperparameters for both stages are summarized in tables I and II.

We pretrain on the full dataset mixture described in Section IV-D. During training, we first resize images to 256x342, then randomly crop and apply color jitter, which yields 224x224 images. We train for 48k steps with a global batch size of 2560 with a constant learning rate of

3e-4. The vision encoder uses a learning rate one tenth that of the rest of the model.

We finetune the pretrained policy on demonstrations from individual tasks. We found that the optimal checkpoint generally occurred earlier in training for simulated tasks than for real tasks. As a result, we finetune for 30k steps for real tasks and for 10k steps for simulated tasks with a global batch size of 320 and a reduced learning rate of 2e-5. We leave co-training, alternate learning rate schedules, and strategies for selecting optimal pretraining and finetuning checkpoints to future work. During finetuning, we use the same image augmentation hyperparameters as during pretraining.

While we compute the loss on 16 action steps during training, during deployment we only execute eight timesteps before recomputing actions [1]. As in training, images are resized to 256x342, but then center-cropped to 224x224 rather than randomly cropped. The policy loop executes at a rate of 10 Hz.

C. Experimental Details

1) Platform: We focus here on tabletop bimanual manipulation using two Franka FR3 robot arms with parallel gripper and TRI’s finray-style fingers [85]. See Figure 10, top-left, for an overview of the physical platform. We had one major hardware upgrade that changed the gripper model and wrist camera configurations; see Sec. VII-A for more details. We collected training data on both platforms, and all real-world evaluations presented here are done on the new platform. We used a total of nine robot stations, see details in Table S1. We run the Franka robots using our custom joint impedance controller, with a differential inverse kinematics controller on top to translate end-effector relative SE(3) commands [48] from the human teleoperator or the LBM policy. The differential inverse kinematics controller also handles collision avoidance. For the new platform, we use the WSG50-110 gripper on each arm. The workspace contain two FRAMOS D415e scene cameras and each wrist has two FLIR Blackfly S BFS-PGE-23S3C-CS cameras. All policies are run at 10 Hz.

2) Simulation: Our primary usage for simulation is for evaluation. One of the main bottlenecks for iterating on policy design is evaluation both in terms of throughput and level of control. Relying on real-world testing alone for enough samples to support any meaningful statistical analysis is prohibitively expensive and time consuming. To address this, we have been developing our simulation benchmark, `lmb_eval`, which is built on top of Drake [86]. `lmb_eval` uses curated assets and human-authored scenarios and tasks. Each simulation is deterministic given a random seed (however, the GPU-based policies do not have the same guarantee).

Our simulation policy evaluations are run against four scenarios, all of which differ in their affordance for

task complexity, object types, and object count (see Figures S2, S3 S4 and S5). All scenarios contain a workspace modeled after a real-world robot station (see station details in Table S1). All four scenarios are inspired by kitchen settings and focus on food preparation and organization. All four scenarios contain task-relevant manipulands (objects that are manipulated as part of the task) in addition to varying numbers of distractors. The first two scenarios are modeled after the old hardware platform, and the later two are modeled after the new hardware platform.

The **DryingRack (D)** scenario contains plates, mugs, and spatulas, in addition to a mug holder, utensil crock, and a dish drying rack. This scenario has 13 tasks. The **Shelf (S)** scenario contains various types of fruits and vegetables, a shelf, fruit bowl, bin, cereal box, and a cutting board. There are 12 tasks in this scenario. The **Breakfast (B)** scenario contains various types of fruits, a mug, plate, coaster, and a cup. This scenario is designed specifically for testing language conditioning since the initial conditions for different tasks visually appear the same. There are 18 tasks in this scenario. The **Kitchen (K)** scenario contains bins, various types of fruits and vegetables, and a plate. Scenario K has 5 tasks, and is the only scenario that is entirely absent in the pretraining dataset, as described in Section IV-D. All tasks in this scenario are long horizon, and some require nonprehensile manipulation or understanding of semantic attributes (e.g., vegetable vs fruit, large vs small). Scenarios B and K are somewhat simpler in terms of scene complexity as compared to the first two scenarios. A fifth scenario (**0**) is used for debugging policies early on during training and consists of just one task where the robot has to move a box to the center of the table. This scenario is not used as part of evaluation, but its demonstration data is included in the pretraining dataset.

D. Pretraining data

Our pretraining mixture, called **Ramen**, consists of a large-scale dataset of robot demonstrations totaling ~1695 hours of demonstration, including high-quality data collected at TRI (~545 hours; **TRI-Ramen**) combined with curated external robot data (~1150 hours; **OXE-Ramen**).

TRI-Ramen data consists of a total of 545 hours of real data over 532 tasks for a total of 64,262 demonstrations. This is made up of: **TRI-Ramen-Real** - 468 hours, 362 tasks and 46063 demonstrations collected across 9 hardware stations; **TRI-Ramen-Sim** - 45 hours, 41 tasks and 7348 demonstrations collected across 2 simulation stations; and **TRI-Ramen-UMI** (32 hours, 129 tasks, 10851 demonstrations) collected with the Universal Manipulation Interface [48] using 7 pairs of handheld devices in “in-the-wild” environments.

Hyperparameters	Pretraining	Finetuning (Real)	Finetuning (Sim)	Single-Task (Real)	Single-Task (Sim)
Learning Rate	3e-4	2e-5	2e-5	2e-5	2e-5
Global Batch Size	2560	320	320	320	320
Steps	48000	30000	10000	100000	25000
LR Schedule	Constant	Constant	Constant	Constant	Constant

Table I: Hyperparameters used during pretraining, finetuning, and with single-task models.

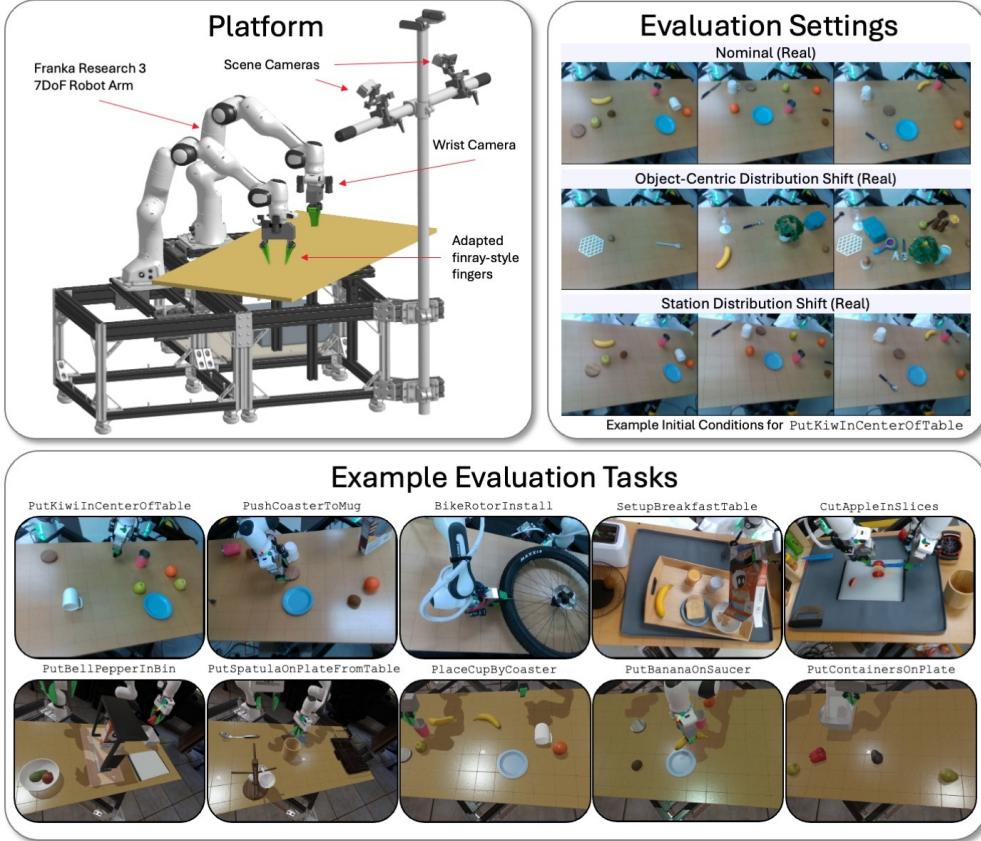


Figure 10: **Experiment Overview.** To study the performance of LBMs, we conduct extensive experiments in both simulation and the real world on a bimanual table-top setting (top left), reporting results under various types of distribution shifts such as object-centric and station (top right). Our evaluation suite includes a diverse array of tasks, including short tasks such as *PutKiwiInCenterOfTable* and long horizon multi-step dexterous tasks such as *CutAppleInSlices* (bottom). The diversity of our evaluation suite allows us to study LBMs under a wide range of conditions. Each column in the top right figure shows the same indexed initial condition under three types of real-world settings. For a full description of the different evaluation settings including simulation (not pictured), refer to Sec. IV-E. The platform varies slightly from the image shown in this figure for different scenarios; see Sec. IV-C1 for an overview. Representative images of example tasks are not all from the same relative scene camera.

TRI-Ramen-Sim tasks used in the pretraining set exclude all five tasks from scenario K , and one task each from scenarios D , S and B ; results of evaluating these “unseen” tasks can be found in Section III-B.

For each of the 40 simulation tasks from scenarios D , S and B that are part of the pretraining set, we collect corresponding real-world demonstrations of the same task in the similar environments (subject to hardware differences across fleet). Scenario B tasks are collected with matched initial conditions as in simulation. Scenarios D and S are collected with manually created initial

conditions. These are part of **TRI-Ramen-Real** (i.e., they are also part of the pretraining set). During real-world data collection, we distributed each task’s demonstrations evenly across 2 to 4 robot stations, such that each station produces approximately 100 demonstrations. For each task, one of the stations used to collect real data is the robot station the simulated environment is modeled after.

OXE-Ramen data is a subset of the OpenX-Embodiment datasets. The subset was chosen based on a set of heuristics such as object and environment diversity and total number of episodes. The observations and actions

Hyperparameters	Values
Image Augmentation	
Resize (HxW)	256x342
Random Crop (HxW)	224x224
Brightness Range	[0.8, 1.2]
Contrast Range	[0.6, 1.4]
Saturation Range	[0.8, 1.2]
Hue Range	[-0.05, 0.05]
Model Architecture	
DiT Layers	8
DiT Embedding Dimension	768
CLIP Image Encoder	ViT-B/16

Table II: Image Augmentation and Model Architecture Hyperparameters

were mapped to conform to the **TRI-Ramen** data format. Mappings include standardization of frames of reference and units for end-effector poses and gripper widths, and resizing and cropping images.

During training, we batch balance the **Ramen** datasets by a set of empirically found weights to ensure each batch contains samples of all the datasets. See Table S7 for datasets used in pretraining and associated weights.

1) *Observation and Action spaces*: The observation space includes i) end-effector poses w.r.t. the station base frame (table center), ii) end-effector poses w.r.t. the other end-effector , iii) continuous gripper width, iv) 6 RGB images (missing cameras are zero-padded), and v) one natural-language instruction. The action space includes i) end-effector poses w.r.t. the station’s base frame, and gripper widths. Orientation is represented as a 6D vector that corresponds to the top 2 rows of the rotation matrix. For observations and actions, we use a similar relative trajectory representation as in [48]. Additionally, we use a history of observations ($n_{obs} = 2$) and an action prediction horizon ($n_{horizon} = 15$) as in [1]. Unimanual data from **OXE-Ramen** was converted into bimanual by zero padding the missing arm and randomly swapping the arm’s side. Each episode in **TRI-Ramen** contains a list of language instructions that are randomly sampled during inference. Such a list includes one instruction written by a human and five instructions generated by prompting a LLM (ChatGPT) to give alternative versions of the human-generated instruction. Missing language annotations in **OXE-Ramen** were filled with a generic text: “do something useful”.

2) *Data Normalization*: For data in **Ramen**, normalization is done on a per-feature dimension (e.g., end-effector pose) and per-timestep (i.e., observation history, and action prediction horizon) basis. Values are normalized to fall within a fixed range of $[-1.5, 1.5]$, by being scaled by the the 2nd and 98th percentiles, $x^{0.02}$ and $x^{0.98}$, and being clipped beyond the range $[-1.5, 1.5]$. For all data samples $x_i \in \mathcal{D}$, we compute the corresponding

normalized value y_i :

$$y_i = \min \left(\max \left(-1.5, 2 \frac{x_i - x^{0.02}}{x^{0.98} - x^{0.02}} - 1 \right), 1.5 \right)$$

This shifts and scales the percentile range of 2 to 98 to lie from -1 to 1 while retaining some outliers, but keeps most of the resolution in the high-density center of the data distribution. Since we represent actions relative to the current time’s observations, actions further into the future have a wider spread than the immediate next actions. Computing normalization parameters for each timestep independently better preserves resolution for near-future actions, which are the more important parts to predict accurately. We avoid this normalization procedure for the 6D rotation in the poses to avoid corrupting the rotation matrix. Note that 6D rotation lies in the $[-1, 1]$ range.

The normalization parameters ($x^{0.02}$ and $x^{0.98}$) were calculated independently per data source for **OXE-Ramen** and **TRI-Ramen-UMI**. **TRI-Ramen-Real** and **TRI-Ramen-Sim** are used together to compute **lbm_robot** normalization parameters. At training time, individual datagrams are normalized separately based on their data source. At test time, the **lbm_robot** normalizer is used to de-normalize actions for our robots. Due to an error in the code, some datagrams were normalized incorrectly (with normalization parameters belonging to a different data source) within each batch during pretraining of the models presented in Section III. Due to the cost of performing extensive real-world evaluation, we did not repeat the experiments in Section III, but rather performed a smaller scale experiment on “seen” tasks in simulation to assess the impact of the incorrect normalization. Fig. S21 shows that the difference is small under nominal conditions, and the LBM with the correct normalization parameters performs better under distribution shift.

3) *Dataset filtering*: The TRI-Ramen dataset consists of a number of low-motion frames at the start of certain demonstrations. This is due either to operator error or to the teleoperation UI being loaded more slowly than the start of the demonstration logging. We implemented a simple filtering operation, defining a motion threshold to capture when the gripper moved either more than 5 cm in translation or 15 deg in rotation with respect to its starting pose. We then remove all data from the start of the demonstration until the motion threshold is met.

In simulation, we analyze the effects of filtering out this data and find that, when trained with unfiltered data, the single-task and LBM policies exhibit difficulty to initiate motion at the beginning of each rollout. The severity of this symptom is policy, task and evaluation conditions (nominal vs distribution shift) dependent. Filtering the low motion data improved single-task performance in simulation; however, it led to a surprising decrease in performance for pretrained LBM, where we observed

that it would commit to some uncommanded task more often than before. We therefore made the design choice to pretrain with unfiltered data, but finetune LBMs and train single-task policies using the filtered dataset in simulation tasks. We performed an analysis of training exclusively with filtered data, and our findings are shown in Section XII. Due to the cost of real-world evaluation, we did not study this phenomenon on real world tasks, using the unfiltered version of task-specific finetuning data.

E. Evaluation Tasks

There are two types of tasks for evaluation, “seen” and “unseen”, depending on their presence in the **TRI-Ramen** dataset during pretraining. For simulation evaluation, we selected 16 “seen” tasks randomly from **TRI-Ramen-Sim** and 8 “unseen” tasks, all part of `lmb_eval`. For real-world evaluation, we selected 3 “seen” tasks and 5 “unseen” tasks, where the 3 “seen” tasks are part of the simulation “seen” tasks, i.e., they have matching simulation and real data demonstrations, as described in Section IV-D. The task names and number of demonstrations collected are listed in Tables S2 and S5, while example evaluation tasks shown in Figure 10, bottom.

We measure performance under several distinct conditions, including nominal in-distribution conditions, as well as conditions exhibiting distribution shift. For each task, we test 200 initial conditions in simulation and 50 in real world. See Appendix VIII-B and IX-C for sample initial conditions in simulation and in the real world.

1) Simulation: We select a small subset of our simulation tasks to use for evaluation, covering four scenarios. All tasks in scenarios K as well as many in D and S are designed specifically to be visually ambiguous given their initial conditions, requiring policies to rely on language conditioning to determine what actions to execute. We consider two conditions for simulation experiments: **Nominal (Sim)** and **Distribution Shift (Sim)**.

Nominal (Sim): Under this setting, object poses as well as quantities are drawn from predefined distribution with rejection sampling to obey constraints such as no interpenetration. Additionally, in-distribution scene lighting parameters for a single directional light source are also drawn from predefined distributions. Lighting parameters include color (over the HSV color wheel), intensity, and direction. In addition to the single parameterized light source, a fixed environment map provides additional ambient lighting to the scene.

Distribution Shift (Sim): Most of `lmb_eval`’s distribution shift is implemented to test policy robustness against appearance changes. For lighting, we define a secondary directional light source with intensity and direction parameters drawn from shifted distributions. Additionally, the environment map is drawn from a discrete set of twenty choices unseen during training. Scene-camera extrinsics

and intrinsics are also randomized. Finally, alternate textures and colors are sampled for objects as well as the table top. The level of distribution shift for colors and textures varies, depending on the specific scenario. In addition to pure appearance variations, we also introduced random distractor objects in scenario B . More details are presented in Section VIII-B and Table S3.

2) Real: We test model performance on eight real-world tasks. Three are short-horizon tasks for which both real and simulation data was seen at pretraining. The other five tasks are unseen long-horizon, multistep tasks that require sequencing diverse types of manipulation. For example, the *CutAppleInSlices* task (Figure S6b) requires the robot to use an apple corer to core an apple, retrieve a knife from a crock, unsheathe the knife to slice the apple into halves, slice the halves into slices, and finally wipe the knife with a cloth before re-sheathing it and placing it back into the crock. We further design three distinct evaluation conditions³: **Nominal (Real)**, **Station Distribution Shift (Real)**, and **Object-Centric Distribution Shift (Real)**; see Figure. 10 for an example.

Nominal (Real): For tasks that have simulation counterparts, the initial conditions of manipulands and distractors are matched to initial conditions generated in simulation via the overlay described in Sec.IV-A1. For tasks without simulation counterparts, the original demonstrator of the task created a new set of initial conditions, using the original objects. The tasks are then tested on an robot station with training data coverage.

Station Distribution Shift (Real): To test cross-station transfer, we evaluate policies on stations not in the finetuning dataset for the task. For this setting, we use the same initial conditions for objects and distractors as in **Nominal (Real)**.

Object-Centric Distribution Shift (Real): We implement two types of object-centric distribution shift in real experiments: manipulands and distractors. Each key manipuland for a given task is tested on at least five novel instantiations (see Figures 10 and S10). Evaluation scenes feature various levels of distractor clutter, where distractors were sampled from a set of both seen and novel objects. To the best of our ability, we sourced novel manipulands and distractors that we believe to not be present in any of the pretraining data. In order to create such initial conditions, we start by populating an empty scene with task-specific manipulands using their initial conditions sampled in simulation, then gradually adding different levels of clutter.

³To determine axes of distribution shift, we informally evaluated earlier experimental models under a wide variety of distribution shifts. We selected modes that were operationally easy to implement, avoiding modes for which a large number of initial conditions would fail and therefore provide a less informative signal.

V. DISCUSSION AND CONCLUSION

Large Behavior Models move dexterous manipulation away from task-specific engineering and into a scalable and data-driven paradigm similar to recent progress in language and vision. To rigorously quantify the capabilities of current LBMs, we train a series of models on roughly 1,700 hours of heterogeneous demonstration data and analyze their performance on 1,800 blind A/B-style real-world rollouts and over 47,000 simulation rollouts.

We find that finetuning LBMs into task-specific specialists consistently outperforms from-scratch training with a given amount of finetuning data or allow achieving from-scratch-equivalent performance with 3-5x less data required. These differences are also amplified under deployment distribution shift—when test-time conditions differ from those encountered during training. This finding is critical because distribution shift is virtually inescapable in real-world use cases and is often omitted from empirical robotics work, masking important information about real-world utility.

We also find that finetuned performance smoothly improves with increasing pretraining data. At the data scales we examined, we find no evidence of performance discontinuities or sharp inflection points.

Interestingly, we encountered mixed results with non-finetuned LBMs. Encouragingly, we found that a single network is able to learn many tasks simultaneously, but we don’t observe consistent outperformance of from-scratch single-task training without finetuning. We expect this is partially due to language-steering brittleness of our models with their small language encoders. We’ve seen promising early signs that larger VLA prototypes overcome some of this difficulty, but more work is required to rigorously examine this effect in higher-language-capacity models.

Our findings largely support the recent surge in popularity of LBM-style robot foundation models, adding to evidence that large-scale pretraining on diverse robot data is a viable path towards more capable robots. However, we also find evidence for caution in the field. Many of the effects we observe were only measurable with larger-than-standard sample sizes and careful statistical testing that is non-standard for empirical robotics. Due to the size of current effects and the magnitude of experimental noise, there is significant risk that many robotics papers are measuring statistical noise due to insufficient statistical power. Additionally, we find that decisions like data normalization have a large effect on downstream performance, often dominating architectural or algorithmic changes; when comparing methods it is critical that these design choices are studied in isolation to avoid conflating the source of performance changes.

A. Limitations

One important limitation of our analysis is that we do not explicitly account for the stochasticity across training when we compare two policy architectures (this would be very expensive to do with statistical significance). Specifically, given a fixed dataset, fixed (stochastic) evaluation benchmark, and a policy architecture, we have

$$p(\text{success}|\text{dataset}) = \int p_{\text{eval}}(\text{success}|w) p_{\text{train}}(w|\text{dataset}) dw,$$

where w are the parameters (weights) of the policy. Our confidence intervals are computed for the first term, but do not account for the second.

We made a decision to run 50 real-world rollouts per task per policy per condition, and to further reduce the measurement uncertainty with hardware displays for reproducible initial conditions. The reproducible initial conditions did mean that each rollout took more time; we intend to continue to optimize the evaluation protocol to improve throughput. Additionally, despite these experimental protocols designed to minimize environment variability and human error, we expect that both initial condition and scoring mistakes are non-zero, likely adding to the noise of our measurements. As a result, our real-world results potentially miss small-magnitude effects due to signal-to-noise limitations.

We also study LBMs with modestly-sized language encoders pretrained via CLIP. While we expect many of our findings will generalize to larger VLAs, we expect that some aspects like language steerability will differ in that setting.

REFERENCES

- [1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2024.
- [2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, 2023. [Online]. Available: <https://roboticsproceedings.org/rss19/p078.pdf>
- [3] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid, "Aloha unleashed: A simple recipe for robot dexterity," in *8th Annual Conference on Robot Learning*, 2024.
- [4] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," 2024. [Online]. Available: <https://arxiv.org/abs/2405.12213>
- [5] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, " π_0 : A vision-language-action flow model for general robot control," 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [6] NVIDIA, :, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu, "Gr0ot n1: An open foundation model for generalist humanoid robots," 2025. [Online]. Available: <https://arxiv.org/abs/2503.14734>
- [7] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. D. Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenklever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. H. Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou, "Gemini robotics: Bringing ai into the physical world," 2025. [Online]. Available: <https://arxiv.org/abs/2503.20020>
- [8] L. Wang, X. Chen, J. Zhao, and K. He, "Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers," *Advances in neural information processing systems*, vol. 37, pp. 124 420–124 450, 2024.
- [9] 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," Mar. 2025, arXiv:2410.07864 [cs]. [Online]. Available: <http://arxiv.org/abs/2410.07864>
- [10] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2023, pp. 3992–4003.
- [11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.
- [12] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 11 975–11 986.
- [13] M. Oquab, T. Darcret, 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," *Transactions on Machine Learning Research Journal*, pp. 1–31, 2024.
- [14] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutscher, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Lukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kirov, M. Knight, D. Kokotajlo, Lukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph, "Gpt-4 technical report," 2024. [Online]. Available: <https://arxiv.org/abs/2303.08774>
- [15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

- [16] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn, “Droid: A large-scale in-the-wild robot manipulation dataset,” 2024.
- [17] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaaf, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin, “Open x-embodiment: Robotic learning datasets and rt-x models,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.08864>
- [18] T. AgiBot-World, “AgiBot World Colosseo: Large-scale Manipulation Platform for Scalable and Intelligent Embodied Systems.”
- [19] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, “Openvla: An open-source vision-language-action model,” in *8th Annual Conference on Robot Learning*.
- [20] A. Brohan, N. Brown, J. Carbal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.15818>
- [21] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, “ $\pi_{0,5}$: a vision-language-action model with open-world generalization,” 2025. [Online]. Available: <https://arxiv.org/abs/2504.16054>
- [22] J. Yang, R. Tan, Q. Wu, R. Zheng, B. Peng, Y. Liang, Y. Gu, M. Cai, S. Ye, J. Jang et al., “Magma: A foundation model for multimodal ai agents,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 14 203–14 214.
- [23] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg et al., “A generalist agent,” *Transactions on Machine Learning Research*.
- [24] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, “Palm-e: An embodied multimodal language model,” 2023. [Online]. Available: <https://arxiv.org/abs/2303.03378>
- [25] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, “Robotic control via embodied chain-of-thought reasoning,” 2025. [Online]. Available: <https://arxiv.org/abs/2407.08693>
- [26] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang, “On the opportunities and risks of foundation models,” 2022. [Online]. Available: <https://arxiv.org/abs/2108.07258>
- [27] G. R. Team, “Gemini Robotics: Bringing AI into the Physical World,” Tech. Rep., Mar. 2025. [Online]. Available: <https://deepmind.google/discover/blog/gemini-robotics-brings-ai-into-the-physical-world/>
- [28] A. Brohan, N. Brown, J. Carbal, Y. Chebotar, J. Dabis, C. Finn,

- K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, “Rt-1: Robotics transformer for real-world control at scale,” 2023. [Online]. Available: <https://arxiv.org/abs/2212.06817>
- [29] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, “OpenVLA: An Open-Source Vision-Language-Action Model,” Sep. 2024, arXiv:2406.09246 [cs]. [Online]. Available: <http://arxiv.org/abs/2406.09246>
- [30] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.
- [31] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima *et al.*, “The pile: An 800gb dataset of diverse text for language modeling,” *arXiv preprint arXiv:2101.00027*, 2020.
- [32] J. Dodge, M. Sap, A. Marasović, W. Agnew, G. Ilharco, D. Groeneweld, M. Mitchell, and M. Gardner, “Documenting large webtext corpora: A case study on the colossal clean crawled corpus,” *arXiv preprint arXiv:2104.08758*, 2021.
- [33] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman *et al.*, “Laion-5b: An open large-scale dataset for training next generation image-text models,” *Advances in neural information processing systems*, vol. 35, pp. 25 278–25 294, 2022.
- [34] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki, “Laion-400m: Open dataset of clip-filtered 400 million image-text pairs,” *arXiv preprint arXiv:2111.02114*, 2021.
- [35] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” in *NeurIPS*, 2023.
- [36] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, “Bridge data: Boosting generalization of robotic skills with cross-domain datasets,” 2021. [Online]. Available: <https://arxiv.org/abs/2109.13396>
- [37] H.-S. Fang, H. Fang, Z. Tang, J. Liu, J. Wang, H. Zhu, and C. Lu, “Rh20t: A robotic dataset for learning diverse skills in one-shot,” in *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.
- [38] AgiBot-World-Contributors, Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Huang, S. Jiang, Y. Jiang, C. Jing, H. Li, J. Li, C. Liu, Y. Liu, Y. Lu, J. Luo, P. Luo, Y. Mu, Y. Niu, Y. Pan, J. Pang, Y. Qiao, G. Ren, C. Ruan, J. Shan, Y. Shen, C. Shi, M. Shi, M. Shi, C. Sima, J. Song, H. Wang, W. Wang, D. Wei, C. Xie, G. Xu, J. Yan, C. Yang, L. Yang, S. Yang, M. Yao, J. Zeng, C. Zhang, Q. Zhang, B. Zhao, C. Zhao, J. Zhao, and J. Zhu, “Agiobot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.06669>
- [39] H. Geng, F. Wang, S. Wei, Y. Li, B. Wang, B. An, C. T. Cheng, H. Lou, P. Li, Y.-J. Wang, Y. Liang, D. Goetting, C. Xu, H. Chen, Y. Qian, Y. Geng, J. Mao, W. Wan, M. Zhang, J. Lyu, S. Zhao, J. Zhang, J. Zhang, C. Zhao, H. Lu, Y. Ding, R. Gong, Y. Wang, Y. Kuang, R. Wu, B. Jia, C. Sferrazza, H. Dong, S. Huang, K. Sreenath, Y. Wang, J. Malik, and P. Abbeel, “Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning,” April 2025. [Online]. Available: <https://github.com/RoboVerseOrg/RoboVerse>
- [40] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar *et al.*, “Orbit: A unified simulation framework for interactive robot learning environments,” *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.
- [41] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su, “Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.00425>
- [42] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan, “Robogen: Towards unleashing infinite data for automated robot learning via generative simulation,” *arXiv preprint arXiv:2311.01455*, 2023.
- [43] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu, “Robocasa: Large-scale simulation of everyday tasks for generalist robots,” *arXiv preprint arXiv:2406.02523*, 2024.
- [44] 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.
- [45] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, “Rlbench: The robot learning benchmark & learning environment,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.
- [46] A. Wei, A. Agarwal, B. Chen, R. Bosworth, N. Pfaff, and R. Tedrake, “Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.22634>
- [47] A. Maddukuri, Z. Jiang, L. Y. Chen, S. Nasiriany, Y. Xie, Y. Fang, W. Huang, Z. Wang, Z. Xu, N. Chernyaev, S. Reed, K. Goldberg, A. Mandlekar, L. Fan, and Y. Zhu, “Sim-and-real co-training: A simple recipe for vision-based robotic manipulation,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.24361>
- [48] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, “Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots,” 2024. [Online]. Available: <https://arxiv.org/abs/2402.10329>
- [49] M. Seo, H. A. Park, S. Yuan, Y. Zhu, and L. Sentis, “Legato: Cross-embodiment imitation using a grasping tool,” *IEEE Robotics and Automation Letters*, vol. 10, no. 3, p. 2854–2861, Mar. 2025. [Online]. Available: <http://dx.doi.org/10.1109/LRA.2025.3535182>
- [50] S. Kareer, D. Patel, R. Punamiya, P. Mathur, S. Cheng, C. Wang, J. Hoffman, and D. Xu, “Egomimic: Scaling imitation learning via egocentric video,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.24221>
- [51] H. Fang, H.-S. Fang, Y. Wang, J. Ren, J. Chen, R. Zhang, W. Wang, and C. Lu, “Airexo: Low-cost exoskeletons for learning whole-arm manipulation in the wild,” 2024. [Online]. Available: <https://arxiv.org/abs/2309.14975>
- [52] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Hogan, G. Richardson, S. Feng, and B. Burchfiel, “Robot learning as an empirical science: Best practices for policy evaluation,” *arXiv preprint arXiv:2409.09491*, 2024.
- [53] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, “Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations,” 2021. [Online]. Available: <https://arxiv.org/abs/2107.14483>
- [54] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman, C. Finn, and S. Levine, “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning,” 2021. [Online]. Available: <https://arxiv.org/abs/1910.10897>
- [55] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, Y. Zhu, and K. Lin, “robosuite: A modular simulation framework and benchmark for robot learning,” in *arXiv preprint arXiv:2009.12293*, 2020.
- [56] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei, “Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments,” 2021. [Online]. Available: <https://arxiv.org/abs/2108.03332>

- [57] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, “Robothor: An open simulation-to-real embodied ai platform,” 2020. [Online]. Available: <https://arxiv.org/abs/2004.06799>
- [58] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, “Sim2real predictivity: Does evaluation in simulation predict real-world performance?” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, p. 6670–6677, Oct. 2020. [Online]. Available: <http://dx.doi.org/10.1109/LRA.2020.3013848>
- [59] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, “Vr-goggles for robots: Real-to-sim domain adaptation for visual control,” 2019. [Online]. Available: <https://arxiv.org/abs/1802.00265>
- [60] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao, “Evaluating real-world robot manipulation policies in simulation,” *arXiv preprint arXiv:2405.05941*, 2024.
- [61] M. Memmel, A. Wagenvoerger, C. Zhu, P. Yin, D. Fox, and A. Gupta, “Asid: Active exploration for system identification in robotic manipulation,” 2024. [Online]. Available: <https://arxiv.org/abs/2404.12308>
- [62] N. Pfaff, E. Fu, J. Binagia, P. Isola, and R. Tedrake, “Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.00370>
- [63] S. Dasari, J. Wang, J. Hong, S. Bahl, Y. Lin, A. Wang, A. Thankaraj, K. Chahal, B. Calli, S. Gupta, D. Held, L. Pinto, D. Pathak, V. Kumar, and A. Gupta, “Rb2: Robotic manipulation benchmarking with a twist,” 2022. [Online]. Available: <https://arxiv.org/abs/2203.08098>
- [64] A. S. Morgan, K. Hang, W. G. Bircher, F. M. Alladkani, A. Gandhi, B. Calli, and A. M. Dollar, “Benchmarking cluttered robot pick-and-place manipulation with the box and blocks test,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 454–461, 2019.
- [65] M. Heo, Y. Lee, D. Lee, and J. J. Lim, “Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation,” *The International Journal of Robotics Research*, p. 02783649241304789, 2023.
- [66] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura, and Y. Yokokohji, “Benchmarking protocols for evaluating small parts robotic assembly systems,” *IEEE robotics and automation letters*, vol. 5, no. 2, pp. 883–889, 2020.
- [67] N. Khargonkar, S. H. Allu, Y. Lu, B. Prabhakaran, Y. Xiang *et al.*, “Scenereplica: Benchmarking real-world robot manipulation by creating replicable scenes,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8258–8264.
- [68] Y. Bekiroglu, N. Marturi, M. A. Roa, K. J. M. Adjigble, T. Pardi, C. Grimm, R. Balasubramanian, K. Hang, and R. Stolkin, “Benchmarking protocol for grasp planning algorithms,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 315–322, 2019.
- [69] F. Bottarel, G. Vezzani, U. Pattacini, and L. Natale, “Graspa 1.0: Graspa is a robot arm grasping performance benchmark,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 836–843, 2020.
- [70] K. Chatzilygeroudis, B. Fichera, I. Lauzana, F. Bu, K. Yao, F. Khadivar, and A. Billard, “Benchmark for bimanual robotic manipulation of semi-deformable objects,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2443–2450, 2020.
- [71] Z. Liu, W. Liu, Y. Qin, F. Xiang, M. Gou, S. Xin, M. A. Roa, B. Calli, H. Su, Y. Sun *et al.*, “Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 486–493, 2021.
- [72] S. Bauer, M. Wüthrich, F. Widmaier, A. Buchholz, S. Stark, A. Goyal, T. Steinbrenner, J. Akpo, S. Joshi, V. Berenz *et al.*, “Real robot challenge: A robotics competition in the cloud,” in *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, 2022, pp. 190–204.
- [73] G. Zhou, V. Dean, M. K. Srirama, A. Rajeswaran, J. Pari, K. Hatch, A. Jain, T. Yu, P. Abbeel, L. Pinto *et al.*, “Train offline, test online: A real robot learning benchmark,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9197–9203.
- [74] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine, “Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.24278>
- [75] D. Snyder, A. J. Hancock, A. Badithela, E. Dixon, P. Miller, R. A. Ambrus, A. Majumdar, M. Itkina, and H. Nishimura, “Is your imitation learning policy better than mine? policy comparison with near-optimal stopping,” *arXiv preprint arXiv:2503.10966*, 2025.
- [76] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, “Deep reinforcement learning at the edge of the statistical precipice,” *Advances in neural information processing systems*, vol. 34, pp. 29 304–29 320, 2021.
- [77] S. Greenland, S. J. Senn, K. J. Rothman, J. B. Carlin, C. Poole, S. N. Goodman, and D. G. Altman, “Statistical tests, p values, confidence intervals, and power: a guide to misinterpretations,” *European journal of epidemiology*, vol. 31, no. 4, pp. 337–350, 2016.
- [78] H.-P. Piepho, “An algorithm for a letter-based representation of all-pairwise comparisons,” *Journal of Computational and Graphical Statistics*, vol. 13, no. 2, pp. 456–466, 2004.
- [79] T. L. Lai, “Nearly optimal sequential tests of composite hypotheses,” *The Annals of Statistics*, pp. 856–886, 1988.
- [80] B. L. Welch, “The generalization of ‘student’s’ problem when several different population variances are involved,” *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.
- [81] W. Peebles and S. Xie, “Scalable diffusion models with transformers,” 2023. [Online]. Available: <https://arxiv.org/abs/2212.09748>
- [82] J. Song, C. Meng, and S. Ermon, “Denoising Diffusion Implicit Models,” Oct. 2022, arXiv:2010.02502 [cs]. [Online]. Available: <https://arxiv.org/abs/2010.02502>
- [83] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” Dec. 2020, arXiv:2006.11239 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/2006.11239>
- [84] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, “π0: A Vision-Language-Action Flow Model for General Robot Control,” Tech. Rep.
- [85] W. Crooks, G. Vukasin, M. O’Sullivan, W. C. Messner, and C. Rogers, “Fin ray® effect inspired soft robotic gripper: From the robosoft grand challenge toward optimization,” *Frontiers Robotics AI*, vol. 3, p. 70, 2016. [Online]. Available: <https://doi.org/10.3389/frobt.2016.00070>
- [86] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2025. [Online]. Available: <https://drake.mit.edu>

SUPPLEMENTARY MATERIALS FOR
A CAREFUL EXAMINATION OF LARGE BEHAVIOR MODELS
FOR MULTITASK DEXTEROUS MANIPULATION

TRI LBM Team

In the following, we provide additional details that complement the information in the paper. Specifically, we provide the author list as well as roles (Section VI); describe the experimental platforms, both for hardware and simulation (Section VII); provide the simulation (Section VIII) and the real-world (Section IX) evaluation details; add additional analysis and insights complementing the results in Section III (Sections X and XI); and discuss data quality and details (Sections XII and XIII).

VI. AUTHORS AND CONTRIBUTIONS

The authors are sorted alphabetically within each group. Unless otherwise indicated, all authors are affiliated with Toyota Research Institute; email: `firstname.lastname@tri.global`.

First authors are primary contributors and made substantial contributions to the work (policy architecture and training, evaluation, simulation): Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, Naveen Kuppuswamy, Kuan-Hui Lee, Katherine Liu, Dale McConachie, Ian McMahon, Haruki Nishimura, Calder Phillips-Grafflin, Charles Richter, Paarth Shah, Krishnan Srinivasan, Blake Wulfe, Chen Xu, Mengchao Zhang.

Second authors assisted with the work (developing infrastructure, data collection, paper edits and feedback): Alex Alspach, Maya Angeles, Kushal Arora, Vitor Campagnolo Guizilini, Alejandro Castro, Dian Chen, Ting-Sheng Chu, Sam Creasey, Sean Curtis, Richard Denitto, Emma Dixon, Eric Dusel, Matthew Ferreira, Aimee Goncalves, Grant Gould, Damrong Guoy, Swati Gupta, Xuchen Han, Kyle Hatch, Brendan Hathaway, Allison Henry, Hillel Hochsztein, Phoebe Horgan, Shun Iwase, Donovon Jackson, Siddharth Karamcheti, Sedrick Keh, Joseph Masterjohn, Jean Mercat, Patrick Miller, Paul Mitiguy, Tony Nguyen, Jeremy Nimmer, Yuki Noguchi, Reko Ong, Aykut Onol, Owen Pfannenstiehl, Richard Poyer, Leticia Priebe Mendes Rocha, Gordon Richardson, Christopher Rodriguez, Derick Seale, Michael Sherman, Mariah Smith-Jones, David Tago, Pavel Tokmakov, Matthew Tran, Basile Van Hoorick, Igor Vasiljevic, Sergey Zakharov, Mark Zolotas.

Last authors led the project and are responsible for strategic decisions (method, benchmark, paper writing and presentation): Rares Ambrus, Kerri Fetzer-Borelli, Ben

Burchfiel, Hadas Kress-Gazit⁴, Siyuan Feng, Stacie Ford, Russ Tedrake.

VII. PLATFORM DETAILS

Table S1 provides a summary of the hardware and simulated robots used for data collection and evaluation.

A. Robot Hardware

All stations contain a tabletop marked with grid lines (useful for correlating object placement on the real robot stations), two Franka Research 3 arms, and two parallel jaw grippers with custom compliant fingers [85]. During data collection, we performed one major hardware upgrade where we switched the Franka Hand to a Schunk WSG50-110 gripper; replaced the single FRAMOS D435 wrist camera with dual FLIR wrist cameras per arm; and replaced the fingers with shorter ones. We will refer to the platform with the Franka Hand and single D435 wrist camera as the old platform, which has been retired and was not used for any reported real-world evaluation results. Each physical robot has meaningfully different scene camera and robot configurations, both are calibrated weekly during operations. Some controller parameters (such as Cartesian velocity bounds) differ slightly between robots, but these do not impact normal operations.

B. Simulated Robot

We model the simulated robot station based on hardware stations, as can be seen in Fig S1. In general, we did not try to achieve exact sim and real matching. We did a best effort to match the hardware platforms behaviorally (e.g., controllers). When simulating the wide angle lens on the new platform, we first render using a pin-hole model with the calibrated and rectified intrinsics, then apply fisheye distortion. On simulated **riverway**, we intentionally made both the simulated scene and wrist cameras wider angle than on hardware for better observability.

VIII. SIMULATION EVALUATION DETAILS

We present additional detailed information regarding the simulation evaluation as well as the predicates used to measure task completion in this section. Both platforms mentioned in Sec. VII-A are used in the simulated scenarios. In particular, scenario *D* and *S* are modeled after the old hardware station **riverway**. These two scenarios are still actively used for simulation evaluation. Scenario *B* and *K* are modeled after a new station **salem**.

⁴Cornell University, Ithaca, NY 14850, USA. Email: hadaskg@cornell.edu

Station Name	HW / Sim	Platform type	Data collection	Evaluation
cabot	sim	new	Yes	Yes
riverway (sim)	sim	old	Yes	Yes
wood_island	HW	new	Yes	Yes
hersey	HW	new	Yes	Yes
maverick	HW	new	Yes	Yes
ruggles	HW	new	Yes	Yes
salem	HW	new	Yes	Yes
davis	HW	new	Yes	Yes
milton	HW	new	Yes	Yes
wollaston	HW	old	Yes	No
riverway (HW)	HW	old	Yes	No

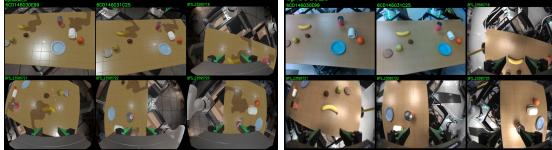
Table S1: Summary of hardware and simulation robots. Data collection and Evaluation columns indicate usage for this paper. Hardware **riverway** and **wollaston** were decommissioned and not used for evaluations.



(a) Rendered camera images on simulated **riverway**.



(b) Camera images on hardware **riverway**.



(c) Rendered camera images on simulation **cabot** station (modified after **salem**).



(a) Nominal (Sim)



(b) Distribution Shift (Sim)

Figure S1: **Camera images from simulated and hardware platforms.** **riverway** is the old platform with 2 wrist cameras and the Franka Hand. **salem** and its counterpart in simulation, **cabot**, belongs to the new platform with 4 wrist cameras and a Schunk gripper.

A. Data used for training or finetuning during simulation evaluation

The amount of training data for each evaluation task is summarized in Table S2. Note that we only used 16 “seen” and 5 “unseen” tasks for simulation evaluation out of a total of 44 tasks, as described in Section IV-D.

B. Simulation initial conditions

Our simulation allows instantiating scenes by drawing from a predefined distribution. For manipulands, for example, we can randomize the number of manipulands, their shape texture and color, and pose relative to any arbitrary frames. The initial condition for any task is implicitly defined as a distribution. At the start of each simulation, a sample initial condition is drawn deterministically based on the simulation seed; when comparing different policies during evaluation, we use the same seed.

Figure S2: **Sample initial conditions** of the *PlaceFruitFromBowlIntoBin* task from scenario S on **riverway**.

We designed the scenarios to be visually ambiguous, meaning policies cannot infer the task solely based on visual appearances. We present sample initial conditions for nominal and distribution shift for a representative task in each scenario in Figures S2, S3, S4 and S5.

The types of distribution shift in simulation used for each scenario are summarized in Table S3.

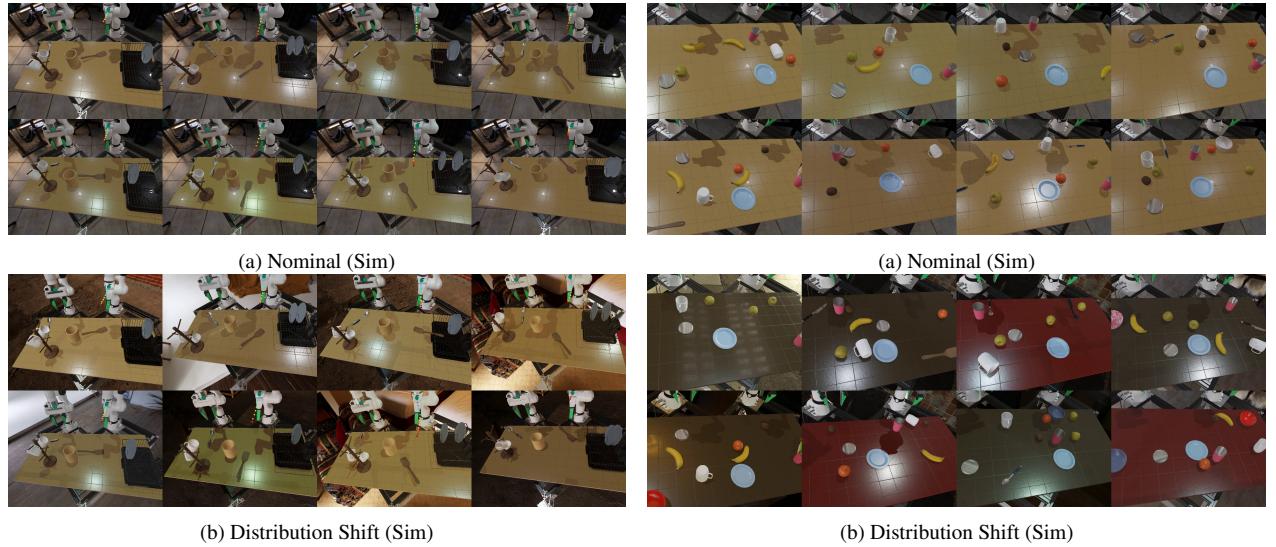
C. Predicates

Success criteria as well as task-completion percentage in simulation are based on task-specific predicates. Each predicate is a function from instantaneous simulation state to a Boolean value, and is logged densely in the raw rollout log. These logged predicate trajectories can be used to perform more granular analysis or answer questions that require stateful information (e.g., did the robot achieve goal A first and then B) offline. Note that the predicates for each task are implemented as part of the task authoring, where the task designer has a nominal strategy in mind.

Task name	Seen in pretraining	Scenario	riverway	cabot	Total
PutSpatulaOnPlateFromDryingRack	Seen	DryingRack (D)	196	0	196
PutSpatulaOnPlateFromTable	Seen	DryingRack (D)	196	0	196
StackPlatesOnTableFromDryingRack	Seen	DryingRack (D)	196	0	196
PutSpatulaInUtensilCrock	Seen	DryingRack (D)	196	0	196
PutSpatulaOnPlateFromUtensilCrock	Unseen	DryingRack (D)	196	0	196
PlaceAppleFromBowlIntoBin	Seen	Shelf (S)	196	0	196
PlaceFruitFromBowlIntoBin	Seen	Shelf (S)	196	0	196
PutBellPepperInBin	Seen	Shelf (S)	196	0	196
StoreCerealBoxUnderShelf	Seen	Shelf (S)	196	0	196
PlaceAvocadoFromBowlIntoBin	Unseen	Shelf (S)	196	0	196
PlaceCupByCoaster	Seen	Breakfast (B)	0	196	196
PushCoasterToCenterOfTable	Seen	Breakfast (B)	0	196	196
PutBananaOnSaucer	Seen	Breakfast (B)	0	49	49
PutMugOnSaucer	Seen	Breakfast (B)	0	196	196
TurnCupUpsideDown	Seen	Breakfast (B)	0	490	490
PutMugInCenterOfTable	Unseen	Breakfast (B)	0	294	294
TurnMugRightsideUp	Seen	Breakfast (B)	0	490	490
PushCoasterToMug	Seen	Breakfast (B)	0	196	196
PutKiwiInCenterOfTable	Seen	Breakfast (B)	0	49	49
TurnLargeContainerUpsideDown	Unseen	Kitchen (K)	0	392	392
PutContainersOnPlate	Unseen	Kitchen (K)	0	392	392
DumpVegetablesFromSmallToLargeContainer	Unseen	Kitchen (K)	0	392	392
PutFruitInLargeContainerAndCoverWithPlate	Unseen	Kitchen (K)	0	392	392
SeparateFruitsVegetablesIntoContainers	Unseen	Kitchen (K)	0	392	392

Table S2: Number of demonstrations for the simulation evaluation tasks per station.

Distribution Shift Type	Scenario S	Scenario D	Scenario B	Scenario K
distractor_textures		✓	✓	
environment_map	✓	✓	✓	✓
lighting	✓	✓	✓	✓
table_top_texture			✓	✓
scene_extrinsics	✓	✓	✓	✓
scene_intrinsics	✓	✓	✓	✓
wrist_intrinsics	✓	✓	✓	✓

Table S3: Types of distribution shift in the simulation results reported in Figures 2 and 6. **distractor_texture** randomizes textures for distractors; **environment_map** samples one of 20 different environment maps; **lighting** adds one more randomized directional light source; **manipuland_texture** applies randomized texture on manipulands; **table_top_texture** applies randomize table-top texture; **scene(wrist)_extrinsics(intrinsics)** applies randomized deltas on top of nominal camera parameters.Figure S3: Sample initial conditions of the *PutSpatulaInUtensilCrock* task from scenario D on **riverway**.Figure S4: Sample initial conditions of the *PushCoasterToCenterOfTable* task from scenario B on **cabot**.

The predicates are thus heavily influenced by the nominal strategy. We analyze task completion only for the five tasks

that belong to the *Kitchen (K)* scenario, as discussed in Section III-B. The task-specific predicates for these tasks

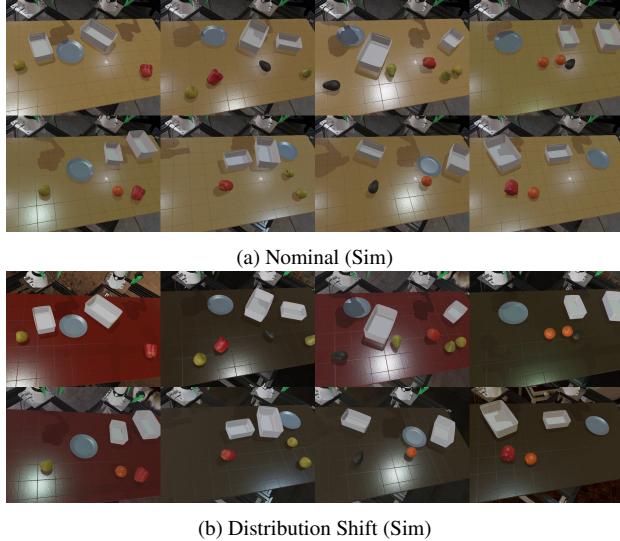


Figure S5: Sample initial conditions of any task in scenario *K* on **cabot**. All five tasks (*SeparateFruitsVegetablesIntoContainers*, *PutContainersOnPlate*, *DumpVegetablesFromSmallToLargeContainer*, *TurnLargeContainerUpsideDown* and *PutFruitInLargeContainerAndCoverWithPlate*) in this scenario have exactly the same distribution for initial conditions.

are shown below:

TurnLargeContainerUpsideDown

- Robot's right hand fingers touched the larger container
- Robot's left hand fingers touched the larger container
- Robot turned the large container over 45 degrees
- Robot turned the large container turn 90 degrees
- Robot turned the large container turn 135 degrees
- Robot flipped the large container upside down

PutContainersOnPlate

- Robot lift the larger container
- The large container on top of the plate
- Robot lift the smaller container
- The smaller container inside the larger container

PutFruitInLargeContainerAndCoverWithPlate

- Robot picked up any fruit
- Any fruit in the large container
- Robot picked up the plate
- All fruit in the large container and the plate on top of the large container

SeparateFruitsVegetablesIntoContainers

- Robot picked up any fruit
- Any fruit in the small container
- Robot picked up any vegetable
- Any vegetable in the large container
- All fruit in the smaller container and all vegetable in the large container

DumpVegetablesFromSmallToLargeContainer

- Robot picked up any vegetable
- Robot put any vegetable in small container
- Robot lifted the small container
- Robot turned the small container over 45 degrees
- Robot turned the small container over 90 degrees
- Robot moved the small container over large container
- Any vegetable inside the large container
- All vegetable inside the large container while robot held the smaller container

D. Missing simulation rollouts

We run every simulation task 200 times per policy and per condition; since the evaluation is run in a distributed manner on the cloud, rollout data can be missing due to various reasons (e.g., sync rollout data to the cloud storage failed). Table S4 below lists all the tasks and conditions for which we use less than 200 rollouts in the results (Section III); all other tasks/policy/conditions have 200 rollouts. In terms of overall impact: out of 238 task/policy/condition combinations, 39 have missing rollouts, for a total of 104 rollouts representing 0.2% of the overall simulation evaluation data.

Task	Policy	Condition	Rollouts
Nominal conditions			
PutBellPepperInBin	Single task	-	199
PutBellPepperInBin	LBM	Finetuned	185
PutSpatulaOnPlateFromRack	LBM	Pretrained	199
StackPlatesOnTableFromRack	LBM	Pretrained	198
StoreCerealBoxUnderShelf	Single task	-	199
PushCoasterToMug	Single task	-	199
PushCoasterToMug	LBM	Pretrained	199
SeparateFruitsVegIntoContainers	Single task	Trained with 15% data	199
PutFruitInLargeContainerAndCoverWithPlate	Single task	Trained with 50% data	199
Distribution shift			
PutAppleFromBowlInBin	LBM	Pretrained	198
PutFruitFromBowlInBin	Single task	-	199
PutFruitFromBowlInBin	LBM	Pretrained	198
PutFruitFromBowlInBin	LBM	Finetuned	198
PutBellPepperInBin	Single task	-	197
PutBellPepperInBin	LBM	Pretrained	193
PutSpatulaOnPlateFromRack	Single task	-	195
PutSpatulaOnPlateFromRack	LBM	Pretrained	197
PutSpatulaOnPlateFromRack	LBM	Finetuned	198
StoreCerealBoxUnderShelf	Single task	-	198
PushCoasterToCenterOfTable	LBM	Pretrained	198
PushCoasterToMug	LBM	Pretrained	198
PushCoasterToMug	LBM	Finetuned	199
PutBananaOnSaucer	Single task	-	199
PutKiwiflCenterOfTable	Single task	-	191
TurnCupUpsideDown	LBM	Pretrained	199
PutContainersOnPlate	Single task	-	199
PutContainersOnPlate	LBM	Trained with 15% data	199
PutContainersOnPlate	Single task	Pretrained	199
PutFruitInLargeContainerAndCoverWithPlate	Single task	-	198
PutFruitInLargeContainerAndCoverWithPlate	Single task	Trained with 15% data	199
SeparateFruitsVegIntoContainers	Single task	-	199
SeparateFruitsVegIntoContainers	LBM	Pretrained	199
TurnLargeContainerUpsideDown	LBM	Finetuned with 50% data	199
BimanualPlaceAvocadoFromBowlIntoBin	Single task	-	198
Fractional pretraining			
DumpVegetablesIntoContainer	LBM	TRI-Ramen-25 + FT@ 100	199
DumpVegetablesIntoContainer	LBM	TRI-Ramen-50 + FT@ 15	199
PutContainersOnPlate	LBM	TRI-Ramen + FT@ 100	198
PutFruitInLargeContainerAndCoverWithPlate	LBM	TRI-Ramen-25 + FT@ 15	198
SeparateFruitsVegIntoContainers	LBM	TRI-Ramen-50 + FT@ 50	199

Table S4: Simulation tasks presented in Section III that have less than 200 rollouts listed by policy and condition. LBM FT=Finetuned LBM, LBM=Pretrained LBM, ST=Single task baseline, DS=Distribution shift.

IX. REAL-WORLD EVALUATION DETAILS

In this section, we provide more details about the real-world tasks themselves, as well as information regarding our real-world evaluation process.

A. Real-world evaluation process

Initializing a rollout: For physically setting up the scene to match desired initial condition, we overlay current live camera feed on top of a snapshot of the desired initial condition as illustrated in Fig. 8. In particular, we use homographic projection to canonicalize images from different stations to facilitate using the same initial

conditions among the hardware fleet. This method works well for objects close to the plane (e.g., table surface) used to compute the homographic projection matrix, but has severe artifacts for tall objects. This can confuse operators during manual alignment. Maintaining the same visibility across the robot fleet is also challenging, and can make certain scene configurations not observable across different stations. Finally, the effort it takes to set up a scene is highly correlated with the task and number of objects involved. We have found the following situations to be particularly challenging operationally: 1) many objects in the scene, 2) non-rigid / deformable objects, 3) messy materials (e.g., food, liquid, fine particles), and 4) irreversible actions (e.g., cutting objects in half, mixing ingredients). A few typical failure modes observed in our experiments when setting up the initial conditions are: 1) incorrect object placement; 2) similar but incorrect objects, typically distractor objects; 3) missed objects, typically in scenarios with many objects or due to poor visibility.

Ending a rollout: After starting a rollout, the operator needs to closely monitor progress and terminate according to certain criteria. Due to the effort involved in setting up each real-world rollout, our operators are instructed to only terminate each rollout if the robot 1) succeeded at the task, 2) exhibited dangerous behavior, or 3) made no progress for a while or was stuck in a repetitive loop. These criteria are more reliant on human judgment and forgiving than those used in simulation, which uses an unconditional fixed timeout and boolean checks for success criteria. It provides a robot ample opportunities to make mistakes and recover, and even allows the robot to perform some other tasks and then perform the commanded task. This in fact is part of the reason for the performance differences for the “seen” tasks in simulation and real-world (see Section X-C for more details).

The real-world evaluation process is highly repetitive and laborious, with ample room for human errors. However, our QA analysis indicates an overall low discrepancy rate when success flags and rubric questions were compared to a secondary set (see QA results in Section IV-A3).

B. Real-world evaluation tasks

We designed five long-horizon complex tasks to specifically test the efficacy of finetuning LBMs on downstream complex tasks (see quantitative results in Section III-B). Here, we provide example film strips for three of the most interesting tasks: *BikeRotorInstall* in Fig. S6a, *CutAppleInSlices* in Fig. S6b and *SetBreakfastTable* in Fig. S6c.

We present the number of task-specific demonstrations used to pretrain or finetune our policies, together with the experimental condition (nominal, station distribution shift or object-centric distribution shift) in Table S5.



(a) Frames from a successful rollout for the *BikeRotorInstall* task on **ruggles**. This task requires a high degree of precision, as well as bimanual coordination: while holding the wheel down with one arm, the robot has to place a rotor and a lockring onto the wheel as well as manipulate a tool with which to tighten the lockring for at least one revolution of the tool around the center axle of the wheel.



(b) Frames from a successful rollout for the *CutAppleInSlices* task on **wood_island**. This is the longest horizon task in our benchmark, and it can take up to 3 minutes to complete. The robot has to first use a tool to remove the core of the apple, after which, using a knife, it has to first slice the apple in half, position each half in the workspace and slice at least one half in at least 3 slices. Finally, the robot has to wipe the knife with a cloth, put it in the sheath, and place it back into the utensil crock.



(c) Frames from a successful rollout for the *SetBreakfastTable* task on **hersey**. This task involves sliding open a cabinet door and manipulating objects in the right order: a bowl, a milk cup, a juice cup, a plate, an apple, a banana, a toast, cereal and a spoon. Each object has to be placed on the tray in the right configuration (i.e., the cups should be upright, the cereal has to be poured into the bowl).

Figure S6: Sample frames from successful long-horizon task rollout videos on hardware. All of the above are from LBM policies finetuned on the respective tasks.

Experiment	Task	Nominal	Diff. Station	DS	wood_island	hersey	maverick	ruggles	salem	davis	milton	Total
Seen	PutKiwiInCenterOfTable	X	X	X	0	0	0	0	49	0	0	49
Seen	TurnMugRightsideUp	X	X	X	0	121	0	0	122	122	125	490
Seen	PushCoasterToMug	X	-	X	98	0	0	0	98	0	0	196
Unseen	ClearKitchenCounter	X	X	-	0	0	0	0	0	305	0	305
Unseen	BikeRotorInstall	X	-	-	0	0	0	533	0	0	0	533
Unseen	CutAppleInSlices	X	-	-	496	0	0	0	0	0	0	496
Unseen	CleanLitterBox	X	-	-	0	0	0	0	0	0	260	260
Unseen	SetUpBreakfastTable	X	-	-	0	229	0	0	0	0	0	229

Table S5: Number of demonstrations for real-world evaluation tasks per robot station. The nominal, diff. station (station distribution shift), and DS (object distribution shift) columns summarize the types of experiments we performed for these tasks during the real-world evaluation.

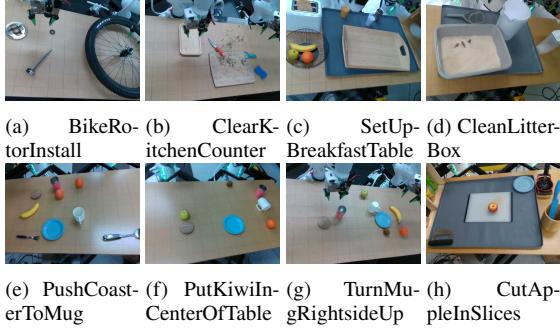


Figure S7: **Initial conditions** for both “seen” tasks (with corresponding quantitative results in Fig. 2) and “unseen” tasks (with corresponding quantitative results in Fig. 4) evaluated on hardware. The subfigure caption denotes the task for which the initial conditions are shown.

C. Sample real-world initial conditions

Fig. S7 shows one initial condition for each real-world task under nominal conditions. To illustrate distribution shift for real-world experiments, we provide sample initial conditions under nominal, station distribution shift, and object-centric distribution shift for *PutKiwiInCenterOfTable* (Fig. S8) and *TurnMugRightsideUp* (Fig. S9).

Fig. S10 shows an overview of the object variation for the real-world experiments, with the objects used under *nominal conditions* shown in the top row and the objects used under *object-centric distribution shift* in the bottom row. Note that for the objects used under distribution shift we vary color, texture as well as shape. The distribution-shift setting is particularly challenging for the tasks involving mugs and coasters, while the tasks involving kiwis have relatively less variation in the manipulands. For the task involving the kiwi, a kiwi was selected randomly from a bin for each IC test bundle.

D. Rubrics for real-world tasks

We present the rubrics used to calculate task completion for the “unseen” real-world tasks. Each question corresponds to a milestone towards task completion and can only be answered with a Yes or No answer. Note that the rubrics are scored manually by human operators. For the corresponding quantitative results, refer to Section III-B.

BimanualClearKitchenCounter

- Robot picked up and placed all tools in the tray receptacle
- Robot picked up the sponge
- Robot held the cutting board in place
- Robot cleaned the cutting board with the sponge
- Robot moved the cutting board aside to make room for cleaning
- Robot swept the waste into the trash bin
- Robot dropped the sponge and returned to the home position

BimanualSetUpBreakfastTable

- Robot opened the cabinet door
- Robot picked the bowl up and placed it on the tray
- Robot placed the milk cup on the tray
- Robot placed the juice cup on the tray
- Are the milk and juice cups upright?
- Robot placed the plate on the tray
- Robot put the apple on the tray
- Robot put the banana on the tray
- Robot picked up at least one toast from the toaster
- Robot put the toast on the plate
- Robot poured the cereal into the bowl
- Robot put the spoon on the tray

BimanualBikeRotorInstall

- Robot picked up rotor
- Robot handed over the rotor to other arm
- Robot placed the rotor on the bike wheel
- Robot seated the rotor onto the bike wheel
- Robot picked up the lockring
- Robot placed the lockring over the rotor
- Robot handed over the tool to other arm
- Robot placed the tool onto lockring
- Robot positioned the tool into lockring grooves
- Robot tightened the lockring at least one revolution
- Robot engaged the threads of the lockring
- Robot took off the tool from the lockring
- Robot puts the tool back onto the table

CutAppleInSlices

- Robot grasped the apple
- Robot grabbed the corer and aligned it with apple center
- Robot cored the apple successfully
- Robot unsheathed the knife and aligned it with the center of the apple
- Robot sliced the apple into two halves
- Robot flipped one half of apple
- Robot flipped the other half of apple
- Robot sliced one half of the apple in at least 3 slices
- Robot picked up the cloth
- Robot wiped the knife
- Robot placed the cloth on top of the shelf
- Robot picked up the knife sheath
- Robot sheathed the knife
- Robot placed the knife into the utensil crock

CleanLitterBox

- Robot grasped the cat litter scoop
- Robot scooped up the cat excrement
- Robot used the other arm to open the trash can
- Robot poured the cat excrement into the trash can
- Robot repeated scooping until all excrement is removed from the litter box
- Robot closed the trash can
- Robot placed the scoop back onto the table

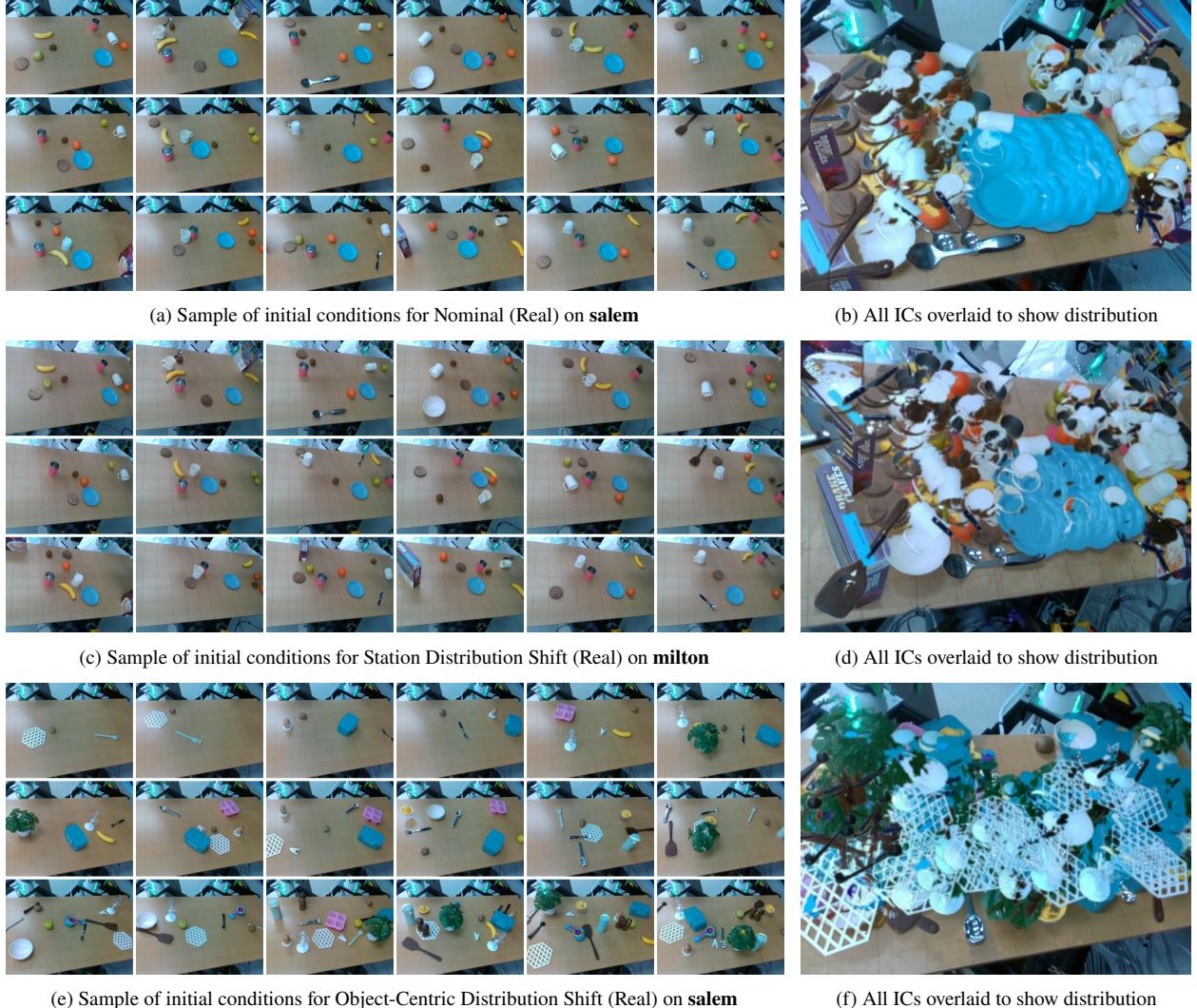


Figure S8: Samples of initial conditions for nominal distribution (a), station distribution shift (c) and object-centric distribution shift (e) for the *PutKiwiInCenterOfTable* task. We present overlays of all initial conditions for each condition (b,d,f).

X. “SEEN” TASKS UNDER NOMINAL CONDITIONS ANALYSIS: BREAKFAST SCENARIO

We provide additional insight into the results of evaluating policies on “seen” tasks during training (Sec. III-A and Fig. 2), specifically in the *Breakfast (B)* scenario.

We note that task success rate correlates with number of demonstrations, i.e., tasks *PutBananaOnSaucer* and *PutKiwiInCenterOfTable* have only 49 demonstrations and low success rate (i.e., single-task success rate of 19.5% and 13.5%, respectively), while *PlaceCupByCoaster* (196 demonstrations) and *TurnCupUpsideDown* (490 demonstrations) have significantly higher success rates (i.e., single-task success rate of 50% and 68.5%). By inspecting many rollouts, we identified two common failure modes: LBMs performing wrong tasks, and policies idling at the beginning of rollouts. To further introspect the per-

formance of our policies on these tasks, we present 1) qualitative results showing the terminal frame of rollouts for each policy (i.e., the last image recorded during a rollout)– this indicates how well a task was executed, and whether there was any confusion with respect to the language command provided; and 2) a quantitative measure of when the first motion was initiated by the policy– this indicates whether the policy started executing the task with a delay, leading to timeouts.

A. *PutKiwiInCenterOfTable* and *PutBananaOnSaucer*

Figures S11 and S12 show the final frames of the policy rollouts, and Figure S13 presents an analysis of the time when the robot’s first motion occurred during rollouts. When inspecting single-task policies’ rollouts, the robot always performs the correct task but often misses the target

(a) Sample of initial conditions for Nominal (Real) **davis**

(b) All ICs overlaid to show distribution

(c) Sample of initial conditions for Station Distribution Shift (Real) on **maverick**

(d) All ICs overlaid to show distribution

(e) Sample of initial conditions for Object-Centric Distribution Shift (Real) on **davis**

(f) All ICs overlaid to show distribution

Figure S9: Samples of initial conditions for nominal distribution (a), station distribution shift (c) and object-centric distribution shift (e) for the *TurnMugRightsideUp* task. We present overlays of all initial conditions for each condition (b,d,f).

manipuland or is unable to place it correctly. Pretrained and finetuned LBM have two common main failure modes: 1) the robot is unable to move away from its starting pose, as seen in Figure S13; 2) the robot performs a different task, as seen in Figures S11 and S12. In both cases, most rollouts end in failure. The finetuned LBM performs better than the pretrained LBM by staying static less often, and by performing the correct task more often.

B. TurnCupUpsideDown

For this task, the finetuned LBM performance is worse than the single-task or pretrained LBM performance. The finetuned LBM often fails to initiate any motion for this task, as seen in Figure S13. This behavior is further investigated in Sec. XII. By inspecting the final frames for this task (Figure S14), we see that the pretrained LBM

tends to execute the wrong task, while the single-task baseline sometimes fails when attempting to grasp the object.

C. Simulation and Real Comparison for PutKiwiInCenterOfTable, TurnMugRightsideUp

These skills are evaluated both in simulation and real-world, with large performance gaps as shown in Figure 2. One of the main reasons is that timeouts are enforced differently in simulation and real evaluations. In simulation, since evaluations are fully automated, timeouts are set with respect to the beginning of each rollout, and are enforced automatically. In real, timeouts (e.g., due to the robot not moving its end effectors) are determined at the discretion of the operators; see Section IX for more details on our real-world evaluation protocol. This difference



Figure S10: Qualitative overview of object variation for real-world **Distribution Shift** experiments. (top) Objects seen during pretraining (bottom). Novel objects used as manipulands or distractors in the **Distribution Shift** setting, which vary in shape and color. Clutter is sampled from the combined set of Nominal and Novel distractors at varying levels. Objects are not pictured to scale and only a representative set of kiwis pictured; a new kiwi was randomly chosen from a larger set for each new IC evaluated.

Task x Policy	Sim	Real	Real w. Sim Timeout
<i>PutKiwiInCenterOfTable</i> x single task	0.135	0.44	0.2
<i>PutKiwiInCenterOfTable</i> x LBM pretrained	0.02	0.2	0.04
<i>PutKiwiInCenterOfTable</i> x LBM finetuned	0.09	0.82	0.7
<i>TurnMugRightsideUp</i> x single task	0.375	0.84	0.46
<i>TurnMugRightsideUp</i> x LBM pretrained	0.485	0.56	0.28
<i>TurnMugRightsideUp</i> x LBM finetuned	0.465	0.88	0.5

Table S6: Success rates for real-world “seen” tasks. Sim and Real columns are the same as reported in Fig. 2. The last column corresponds to applying the simulation timeout to real rollouts.

gives the real policies an advantage in the sense that they can make mistakes and recover. Moreover, this helps the pretrained and finetuned LBM more due to their tendency to perform an incorrect task and then eventually the correct one. After applying the same simulation timeouts to real rollouts (see results in Table S6), the success rates are closer to simulation for *TurnMugRightsideUp*. Further inspecting the rollouts generated when evaluating *PutKiwiInCenterOfTable*, we found that pretrained and finetuned LBM policies in real have almost no visible pauses at the beginning of each rollout, which is in stark contrast to their simulation counterparts (shown in S13). Note that we use the same pretrained LBM for both simulation and real-world evaluation. Finally, we also show side-by-side terminal frames for all policies in both simulation and real-world in Figures S15 and S16.

XI. BAYESIAN ANALYSIS FOR TASK COMPLETION

In Sec. III-B, Figures 4 and 6, for task completion, we depict the full data distribution, the mean and whether the policies are statistically distinct. Figures S17 and S18 correspond to the same data, only here we depict the

uncertainty in the estimate of the true mean of the distribution, used to evaluate whether the policies are statistically distinct.

XII. EFFECTS OF DATASET FILTERING

In this section, we provide additional details regarding the effects of low-motion frames in the training data on policy performance in simulation, as discussed in Section IV-D3. In particular, we experiment with filtering low-motion data at the beginning of each demonstration episode. The filtering process is: for each demonstration, filter out all data points prior to the first timestep that the motion threshold is satisfied. This results in 3.3% **TRI-Ramen-Real** and 11.7% **TRI-Ramen-Sim** data filtered out.

In the following we refer to LBMs trained with filtered data as *filtered-pretrained* LBMs, and contrast them with *unfiltered-pretrained* LBMs. As mentioned in Section IV-D3, we examine the effects of filtering only on the pretraining phase, and finetune LBMs and train single-task baselines using the filtered data.

We observe that the *filtered-pretrained* LBM very quickly commits to a task, whereas the *unfiltered-pretrained* LBM would often take a long time to initiate any motion. However, we encountered an unexpected effect where the *filtered-pretrained* LBM would commit to a different task more often than before, i.e., the policy started suffering from language steerability issues. This significantly reduced performance of the pretrained LBM as shown in Fig. S20, where single-task is statistically better than *unfiltered-pretrained* LBM in 3/16 tasks and 6/16 tasks for *filtered-pretrained* LBM, and finetuned

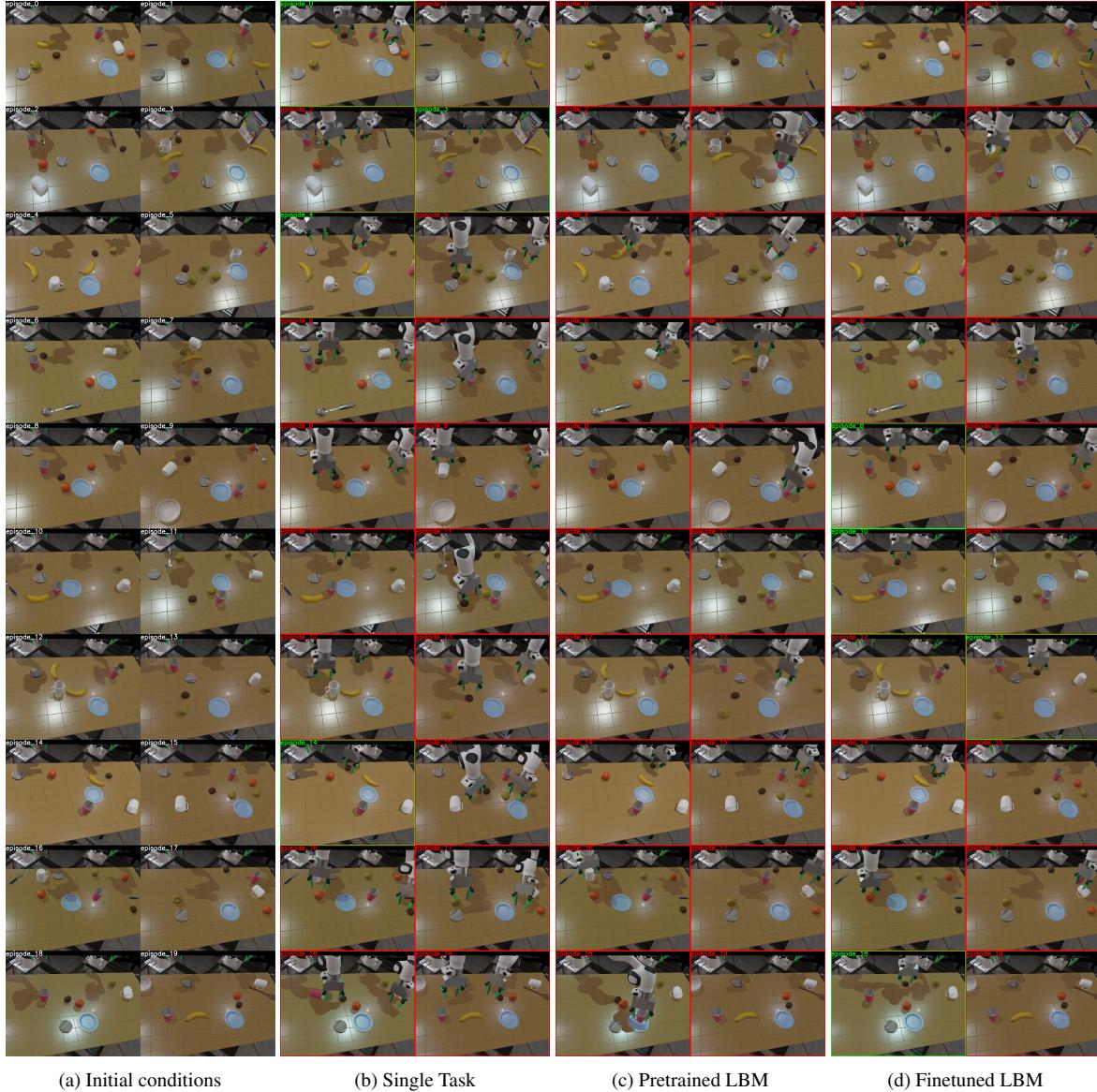


Figure S11: Tiled terminal frames of rollouts from different policies performing *PutKiwiInCenterOfTable* under nominal conditions. The single-task policy often missed the kiwi while grasping. The pretrained LBM often manipulates the wrong object, and sometimes suffers from the inability to move. The finetuned LBM suffers heavily from the inability to move and sometimes performs the wrong task. The green outline around a rollout indicates success, while the red outline indicates failure.

LBM is statistically better than *unfiltered-pretrained* LBM in 5/16 tasks and 9/16 tasks for *filtered-pretrained* LBM.

We further introspected the effect of starting finetuning from either filtered- or unfiltered- pretrained LBMs. We observed that *filtered-pretrained* and *finetuned* LBMs quickly initiate motions, whereas the *unfiltered-pretrained* and *finetuned* LBMs can take excessively long to start any motion. Time-to-motion plots for LBMs are shown in Figure S19. The overall task performance for the *filtered-pretrained* and *finetuned* LBMs are similar to the *unfiltered-pretrained* and *finetuned* models. But for spe-

cific tasks where the *unfiltered-pretrained* and *finetuned* LBMs take excessively long to initiate any motion, the *filtered-pretrained* and *finetuned* LBMs make significant improvements. We provide additional per-task details in Sections X-A and X-B.

Our hypothesis for why filtering low-motion data from the pretraining data caused more severe language steerability issues for our pretrained LBMs is that low-motion data ends up increasing the importance of the language conditioning during training. These data all concentrate at the beginning of each rollout, where scenes are maximally

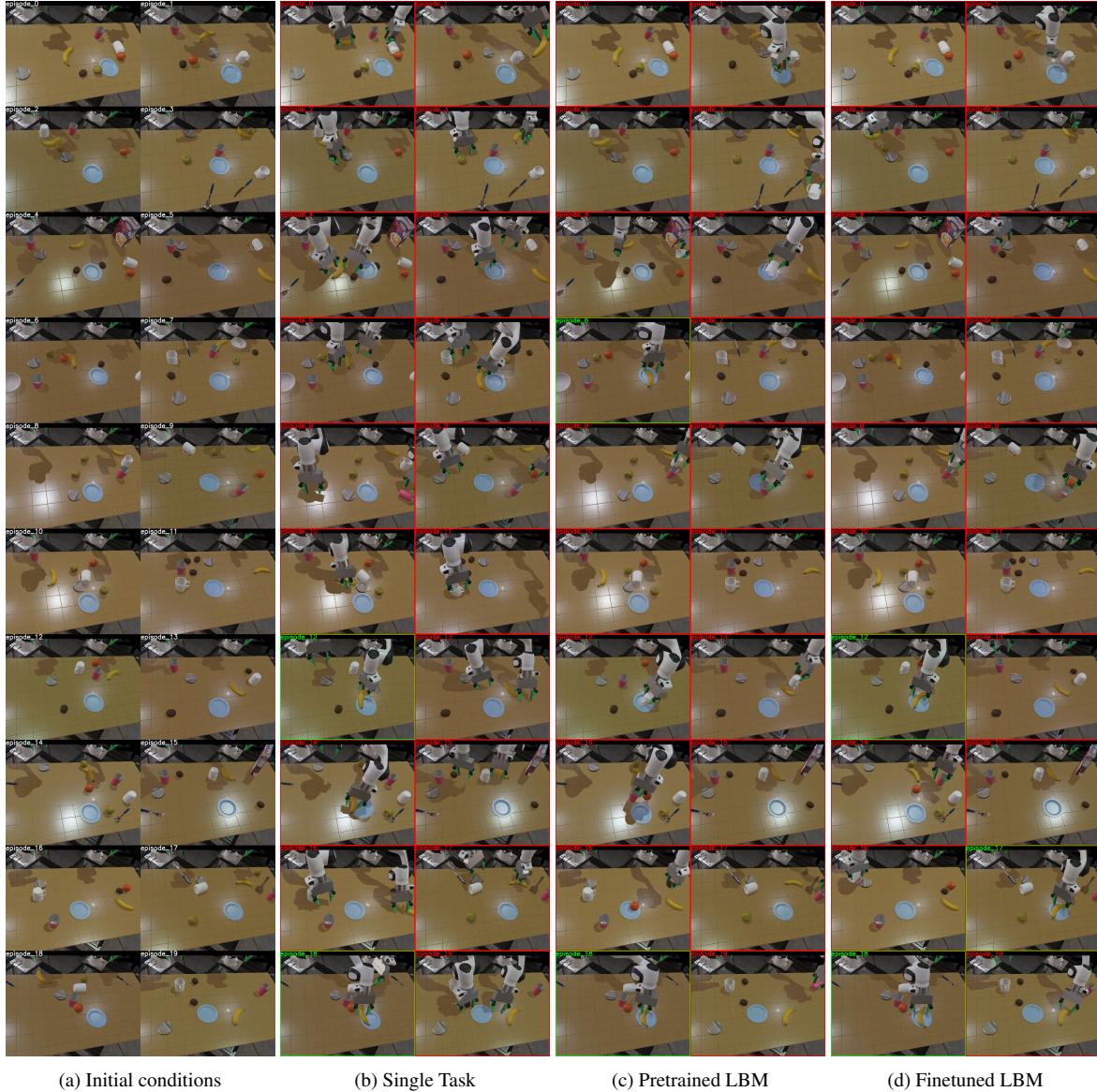


Figure S12: Tiled terminal frames of rollouts from different policies performing *PutBananaOnSaucer* under nominal conditions. The single-task policy often fails to grasp the banana or to place it correctly afterwards. The pretrained LBM’s main failure mode is manipulating other objects. The finetuned LBM suffers heavily from the inability to move. The green outline around a rollout indicates success, while the red outline indicates failure.

visually ambiguous (by benchmark design) and the policy needs to learn to pay close attention to language to perform the correct tasks. See Figure S5 for an example where the starting conditions are the same for five different tasks, and the robot must learn to disambiguate based on language. Another interesting observation is that we have not observed pretrained or finetuned LBMs having difficulties initiating any motions on hardware regardless of filtering. This is likely due to the percentage of low-motion data being much smaller in hardware than in simulation.

XIII. ADDITIONAL PRETRAINING DATASET DETAILS

Table S7 shows detailed information regarding the batch balancing weights used to create the LBM pretraining dataset. This complements Sec. IV-D.

XIV. DATA NORMALIZATION EXPERIMENT

As described in Section IV-D2, we discovered that some of the datagrams in the training data were not normalized correctly. We ran an additional evaluation on “seen” simulation tasks, both in nominal conditions and under distribution shift, to assess the impact of the normalization

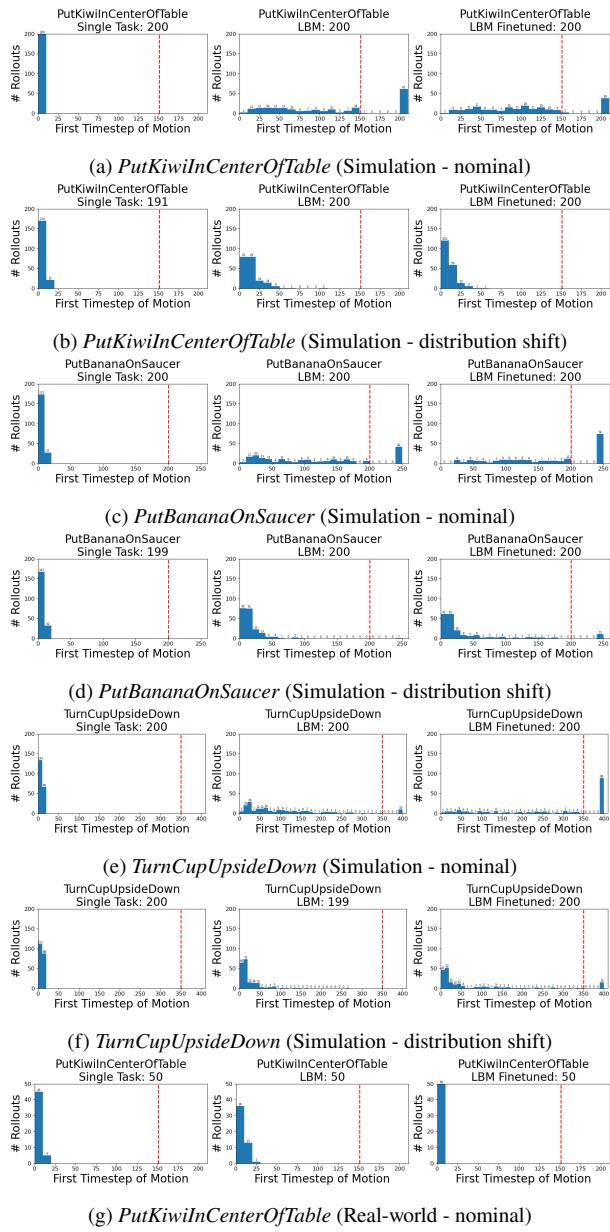


Figure S13: Histogram of robot's first time of motion (defined in IV-D3). Red lines indicate timeout. First column is single-task, second column is pretrained LBM, and third is finetuned LBM. For the three simulation tasks shown here (subplots (a) to (f)), the pretrained and finetuned LBM take significantly longer than single-task to start performing any actions, and in many rollouts, the robot never moved. When LBM policies are tested under distribution shift, they tend to move much sooner than nominal situations. Additionally, in subplot (g) we show the real-world rollouts for the *PutKiwiInCenterOfTable* task to highlight the differences in execution: in real-world, LBMs do not exhibit the same delays before initiating motion as in simulation.

Data source name	Weights
lbm_real	0.5
lbm_sim	0.25
lbm_umi	0.05
bc_z	0.03
berkeley_autolab_ur5	0.007
bridge_data_v2	0.02
droid	0.05
fractal20220817_data	0.03
furniture_bench_dataset_converted_externally_to_rlds	0.02
jaco_play	0.005
language_table	0.015
nyu_franka_play_dataset_converted_externally_to_rlds	0.05
stanford_hydra_dataset_converted_externally_to_rlds	0.01
utokyo_xarm_pick_and_place_converted_externally_to_rlds	0.003
viola	0.005
Total weight	1.045

Table S7: Unnormalized batch balance weights used to train LBM. Weights are normalized to sum to one during training.

error. Figure S21 show how the pretrained LBM with the corrected parameters performs with respect to the rest of the policies. We can see that for these tasks, the error did not significantly affect the performance of the pretrained LBM under nominal conditions; however, for distribution shift, the pretrained LBM without the error outperforms the pretrained LBM with the error on 4/16 tasks and in the aggregate.

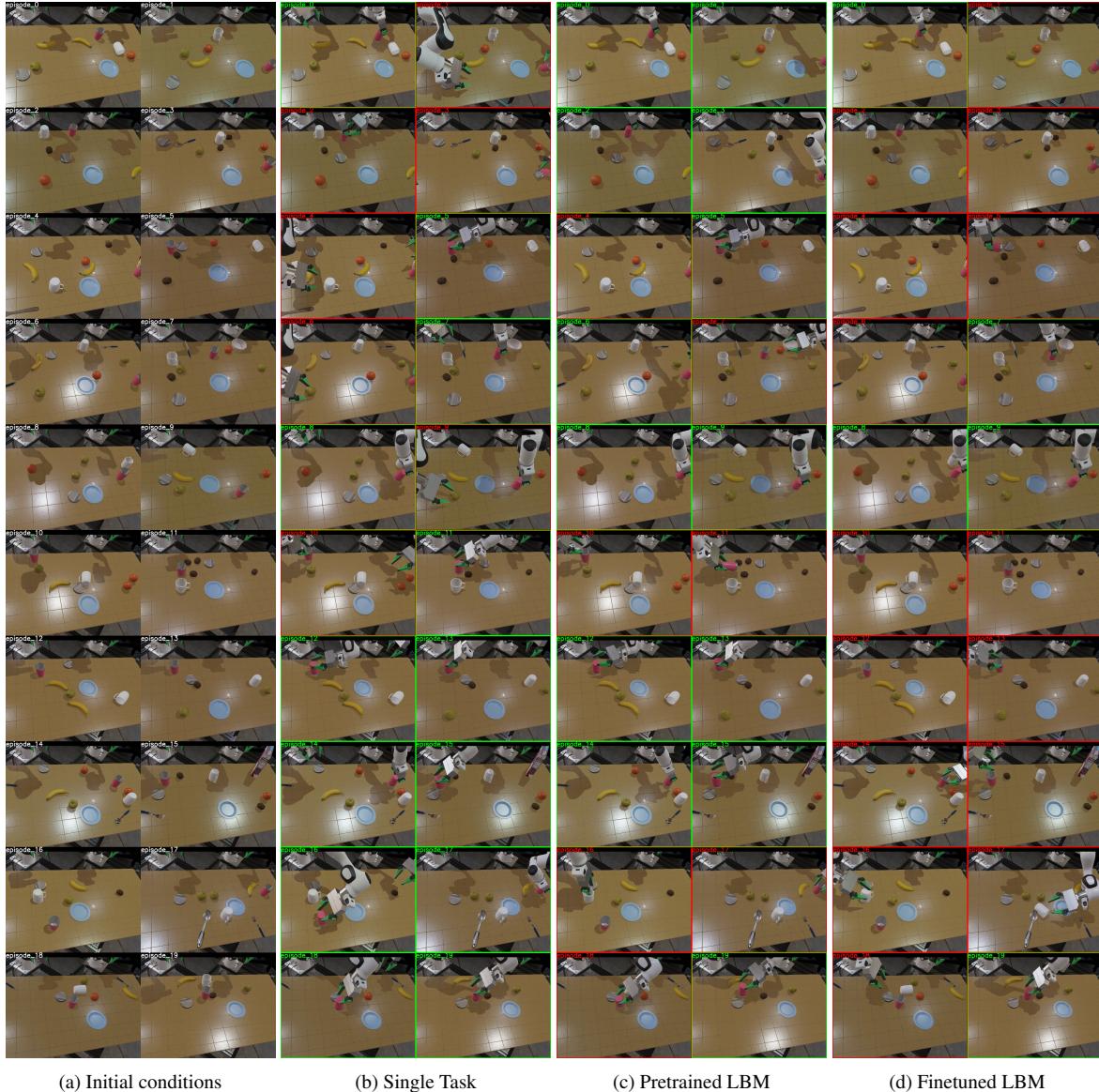


Figure S14: Tiled terminal frames of rollouts from different policies performing *TurnCupUpsideDown* under nominal conditions. The single-task policy performs reasonably, and its main failure mode is when initiating grasps. The pretrained LBM sometimes performs the wrong tasks. The finetuned LBM suffers from the inability to move, further analyzed in Figure S13. The green outline around a rollout indicates success, while the red outline indicates failure.

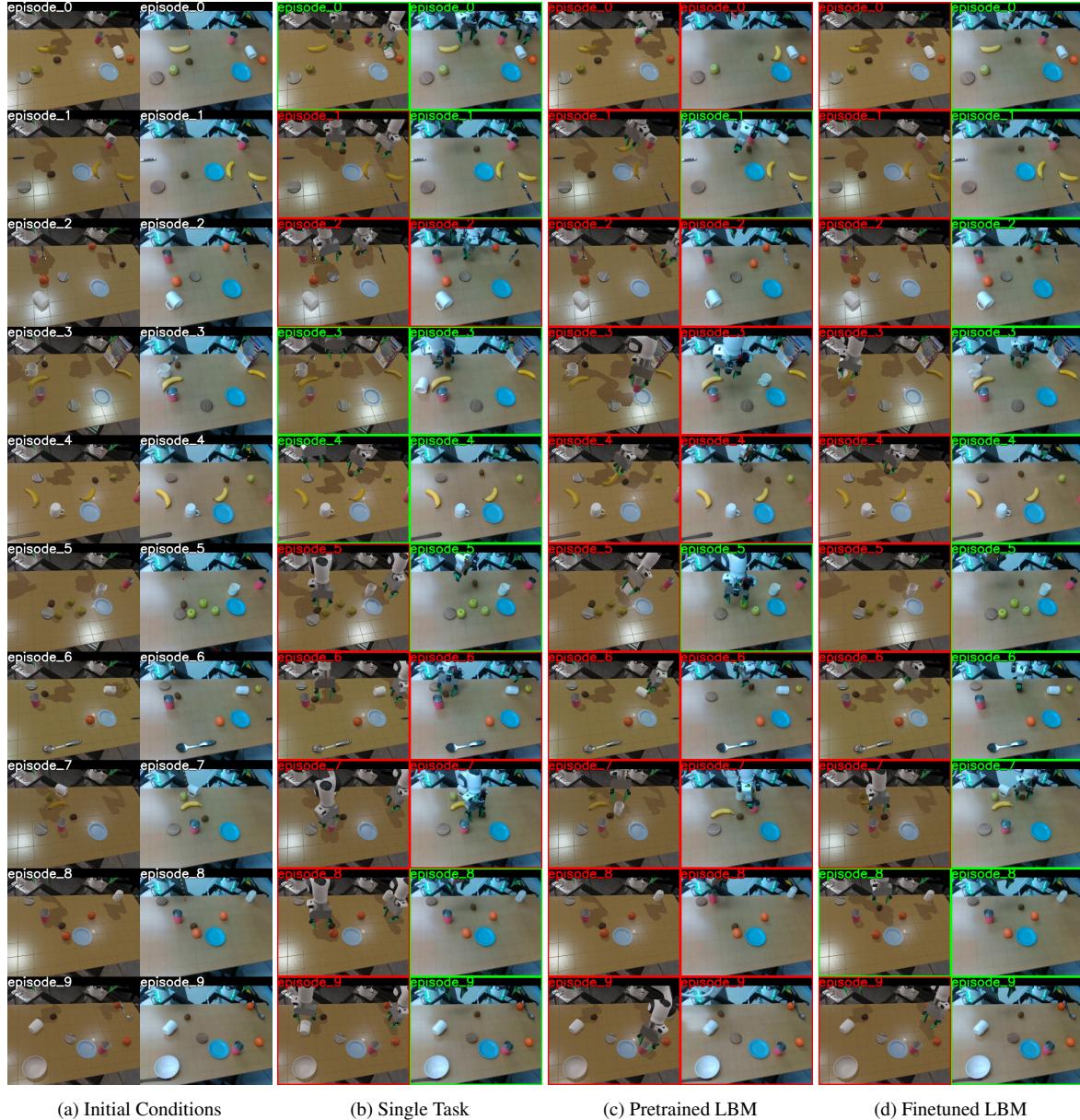


Figure S15: Tiled terminal frames of rollouts from different policies performing *PutKiwiInCenterOfTable* in simulation (left) and on hardware (right) under nominal conditions. For single-task baselines, the policy in simulation is worse at grasping the kiwi compared to on hardware. The pretrained LBM in both simulation and real-world often performs the wrong task (interacting with the task irrelevant objects). The finetuned LBM in simulation suffers more from delays in initiating motion, and performs the wrong task more often than the real counterpart. The green outline around a rollout indicates success, while the red outline indicates failure.



Figure S16: Tiled terminal frames of rollouts from different policies performing *TurnMugRightsideUp* in simulation (left) and on hardware (right) under nominal conditions. The real-world single-task baseline much better than the single-task in simulation due to the more generous timeout protocol for real-world eval (see Section X for more details). The pretrained LBM performs roughly the same in simulation and on hardware. The finetuned LBM performs much worse in simulation due to delays in initiating motion resulting in timeouts. The green outline around a rollout indicates success, while the red outline indicates failure.

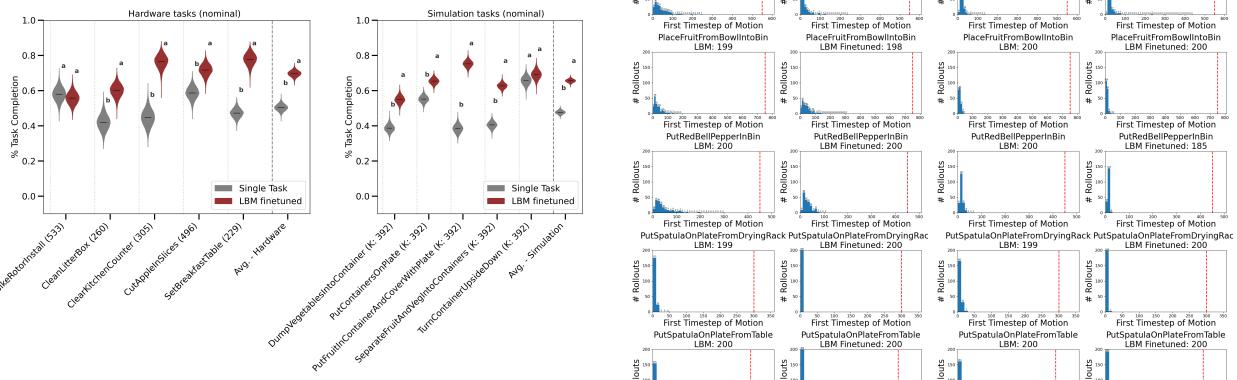


Figure S17: LBM performance on “unseen” tasks in hardware and in simulation evaluated under nominal conditions. We compare the single-task baseline, with LBMs after finetuning. The violin plots represent the Bayesian posterior of the mean Task Completion under a uniform Dirichlet prior. We use statistical hypothesis tests over the mean TC for the CLD letters shown in these plot. These results complement the entire TC data distribution shown in Fig. 4.

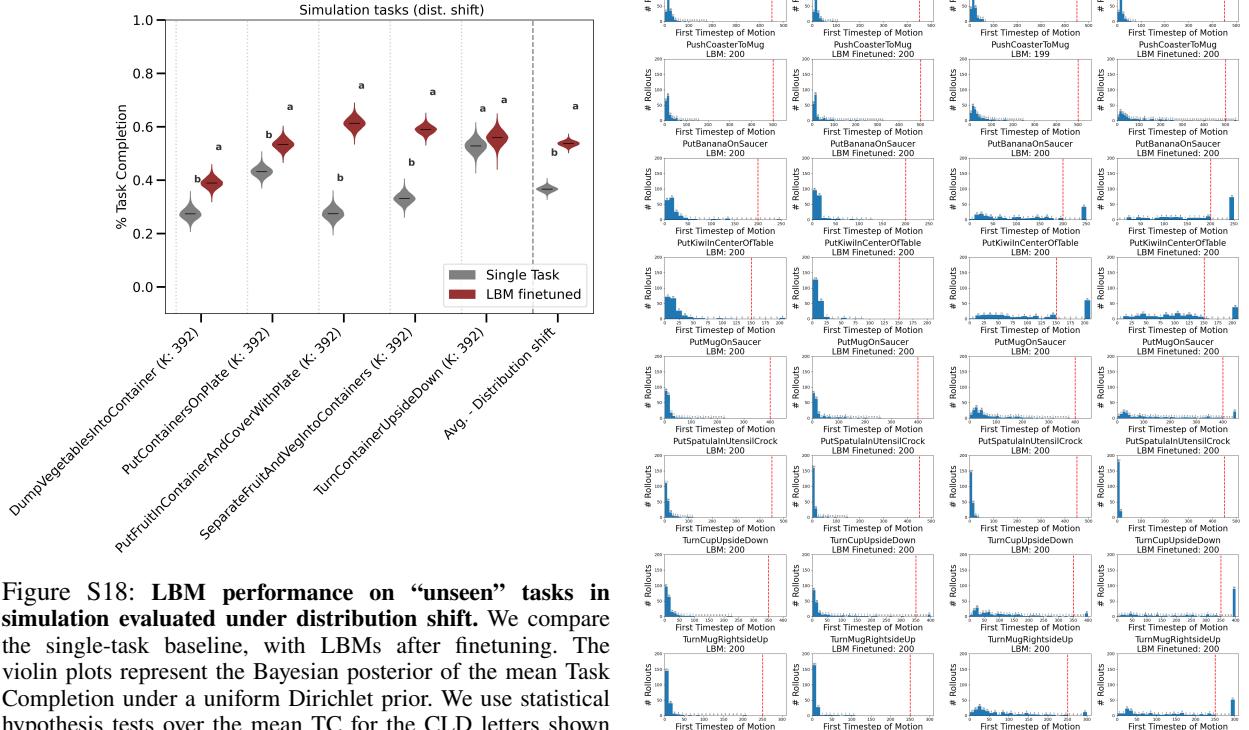
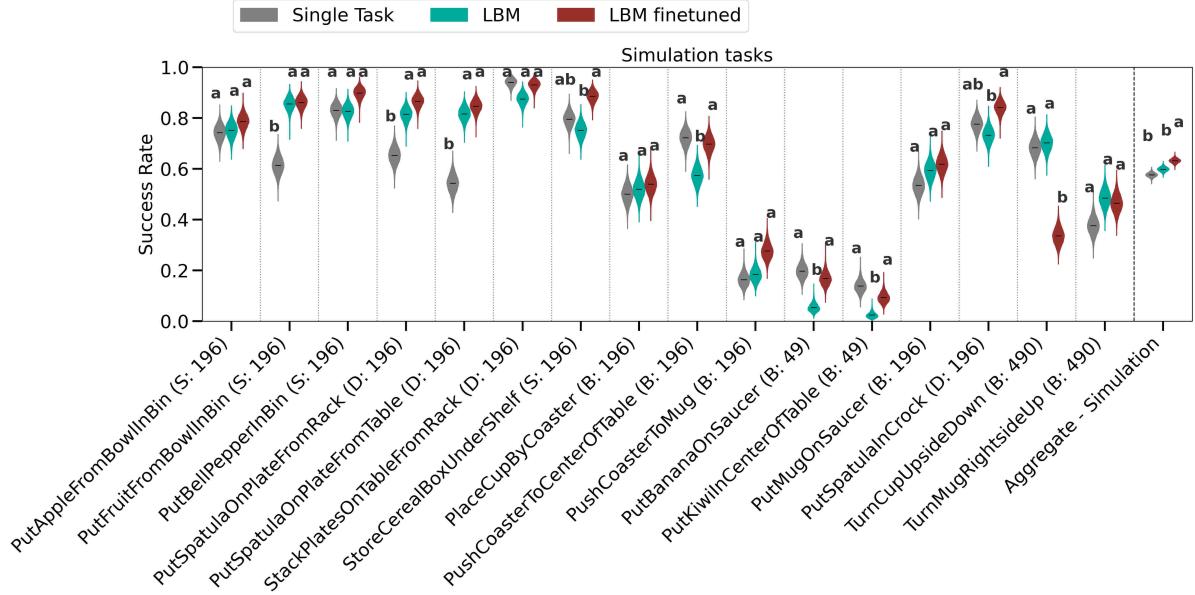


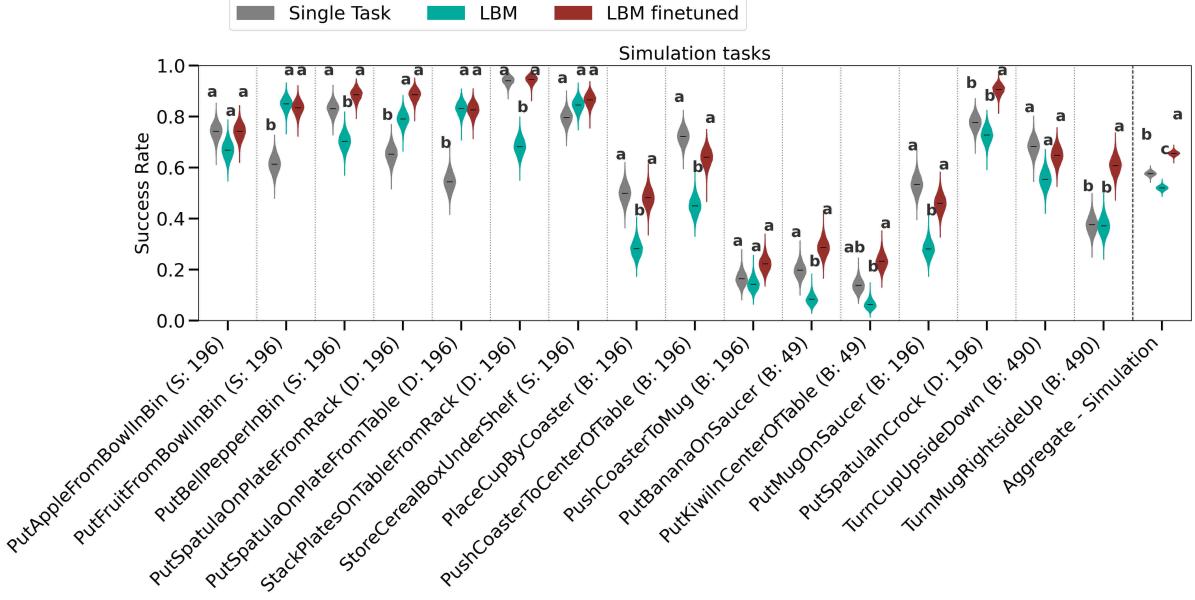
Figure S18: LBM performance on “unseen” tasks in simulation evaluated under distribution shift. We compare the single-task baseline, with LBMs after finetuning. The violin plots represent the Bayesian posterior of the mean Task Completion under a uniform Dirichlet prior. We use statistical hypothesis tests over the mean TC for the CLD letters shown in these plot. These results complement the entire TC data distribution shown in Fig. 6.

(a) Low-motion data filtered for all policies
(b) Low-motion data filtered only for finetuned LBM and single-task

Figure S19: Histogram of robot’s first time of motion (defined in Section IV-D3) for 16 simulation “seen” tasks under nominal conditions. Red lines indicate timeout. Each row corresponds to one task. Each subplot has two columns that corresponds to finetuned and pretrained LBMs respectively. Note that all finetuning are trained without low-motion data. After filtering low-motion data from pretraining data, both pretrained and subsequently finetuned LBMs initiate motions much faster.

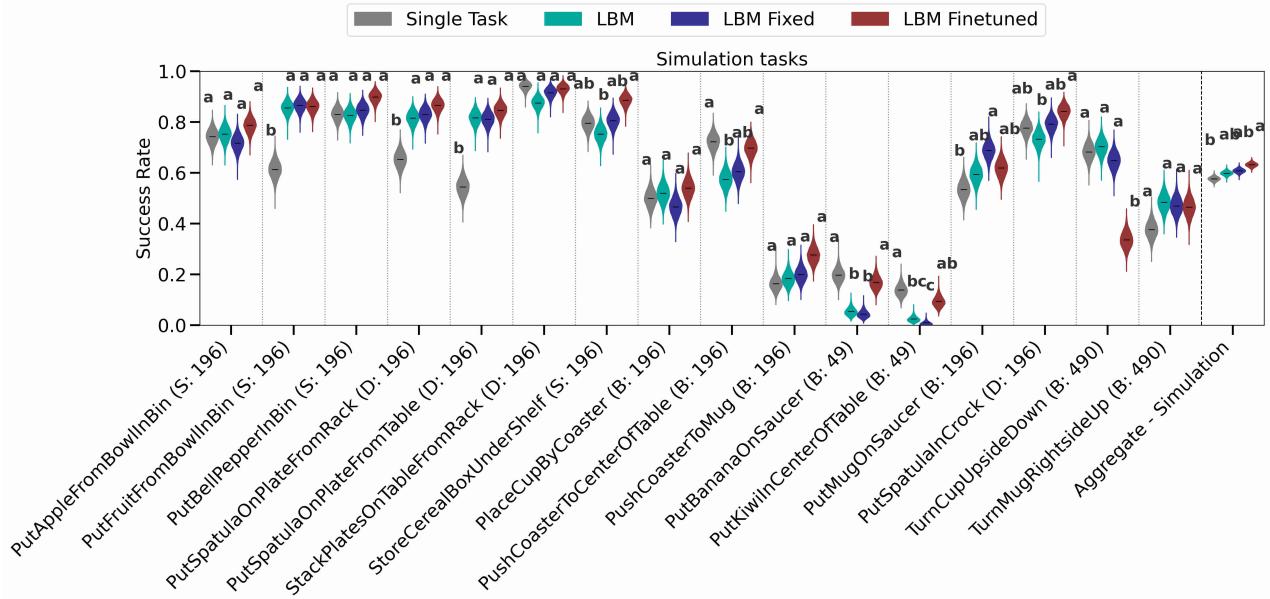


(a) LBM Pretrained using unfiltered dataset, while single-task and finetuned LBM use the filtered dataset.

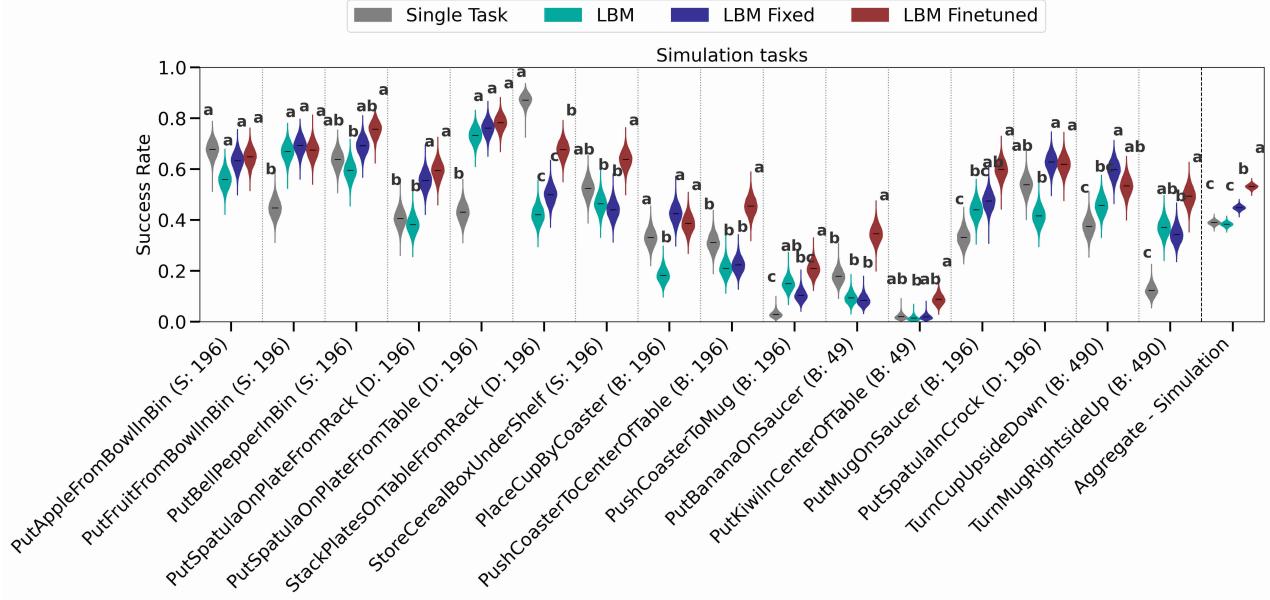


(b) Single-task, pretrained LBM and finetuned LBM all use the filtered dataset.

Figure S20: Analyzing the effect of low-motion data in the pretraining dataset. Finetuning and single-task baseline training were done using the filtered dataset. The evaluations are done under nominal conditions on simulation tasks which are seen during training.



(a) LBM vs LBM Fixed (corrected normalization parameters) evaluated on “seen” tasks under nominal conditions.



(b) LBM vs LBM Fixed (corrected normalization parameters) evaluated on “seen” tasks under distribution-shift conditions.

Figure S21: Comparing pretrained LBMs with and without the normalizer error on “seen” tasks in simulation. The top row is for nominal conditions and the bottom row for distribution shift. This figure adds the corrected pretrained LBM to the results in Figure 2 for easier comparison.