AutoEval: Autonomous Evaluation of Generalist Robot Manipulation Policies in the Real World

Paul Zhiyuan Zhou
Pranav Atreya
You Liang Tan
Karl Pertsch
Sergey Levine

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-42
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-42.html

May 6, 2025

Acknowledgement

Zhiyuan designed and implemented the AutoEval system and Web UI and led the experiments. Pranav assisted in setting up the AutoEval stations and running experiments. You Liang implemented the comparison experiments with SIMPLER and baseline methods and contributed to the AutoEval station setup. Together, all three iterated on reset policy and success detector training. Karl and Sergey provided valuable guidance throughout the project and contributed to the paper writing. We would like to thank Kyle Stachowicz and Mitsuhiko Nakamoto for valuable discussions and feedback on an earlier version of the system.

AutoEval: Autonomous Evaluation of Generalist Robot Manipulation Policies in the Real World

Zhiyuan Zhou[1], **Pranav Atreya**[1], **You Liang Tan**[1,2], **Karl Pertsch**[1], **Sergey Levine**[1]
[1]UC Berkeley, [2]NVIDIA
https://auto-eval.github.io

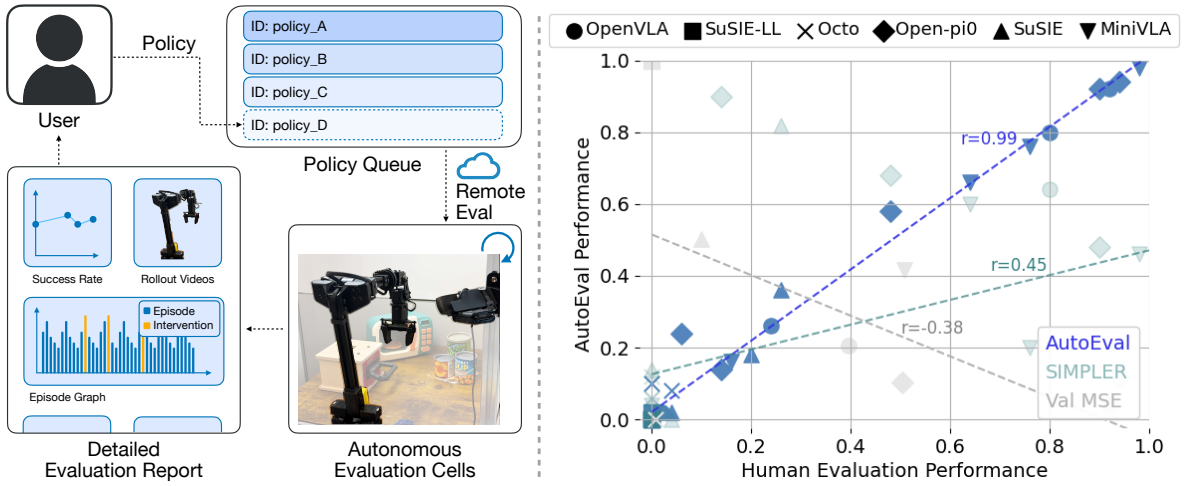

Figure 1: We introduce AutoEval, a system for scalable, automated real robot evaluation of generalist robot policies. Automated evaluation results closely match human-run evaluations, while providing a more reliable performance signal than prior simulated evaluation approaches with photo-realistic environments (SIMPLER) or offline metrics such as validation error (Val MSE). AutoEval reduces human supervision time for evaluation by more than 99%. We provide public access to our AutoEval cells to facilitate standardization and ease of policy benchmarking.

Scalable and reproducible policy evaluation has been a long-standing challenge in robot learning. Evaluations are critical to assess progress and build better policies, but evaluation in the real world, especially at a scale that would provide statistically reliable results, is costly in terms of human time and hard to obtain. Evaluation of increasingly generalist robot policies requires an increasingly diverse repertoire of evaluation environments, making the evaluation bottleneck even more pronounced. To make real-world evaluation of robotic policies more practical, we propose AutoEval, a system to autonomously evaluate generalist robot policies around the clock with minimal human intervention. Users interact with AutoEval by submitting evaluation jobs to the AutoEval queue, much like how software jobs are submitted with a cluster scheduling system, and AutoEval will schedule the policies for evaluation within a framework supplying automatic success detection and automatic scene resets. We show that AutoEval can nearly fully eliminate human involvement in the evaluation process, permitting around the clock evaluations, and the evaluation results correspond closely to ground truth evaluations conducted by hand. To facilitate the evaluation of generalist policies in the robotics community, we provide public access to multiple AutoEval scenes in the popular BridgeData robot setup with WidowX robot arms. In the future, we hope that AutoEval scenes can be set up across institutions to form a diverse and distributed evaluation network.

1. Introduction

Robot foundation models promise to drastically change the robot learning "workflow": instead of training policies for individual tasks or environments, these models are trained across a range of scenes, tasks, and robot embodiments [13, 12, 60, 36, 68, 69, 21, 11, 31], providing generalist policies that can solve new tasks in new settings. This shift to generalist training necessitates an analogous shift in how these policies are evaluated. While traditional evaluations for single-task policies typically involve a few dozen policy rollouts that are

practical to do by hand, robot foundation models may require hundreds of rollouts across a variety of tasks and scenes to obtain an accurate assessment of their generalist capabilities. For instance, a comprehensive evaluation of the recently introduced OpenVLA model [36] against its baselines required more than 2,500 rollouts across four robot setups and three institutions, and a total of more than 100 hours of human labor for resetting scenes, rolling out policies, and recording success rates. Evaluations during the course of model development and design ablations may compound this effort multiple times over. Prior works have tried to address this evaluation bottleneck by building realistic simulated environments for evaluation [46], but the gap between simulation and the real world can render results unreliable, and many tasks like cloth or liquid manipulation are challenging to simulate at sufficient fidelity. In this work we aim to develop a system for robot policy evaluation that combines the *reliability* of real world evaluations, with the *scalability* required for the evaluation of generalist robot policies.

A key bottleneck for the scalability of real-world robot evaluations is the human operator time required to conduct the evaluation, reset the scene, and score policy success. If we can reduce required human involvement to a minimum, we can drastically increase the throughput of real robot evaluations by running evaluations around the clock. To this end, we propose AutoEval, a system for designing *autonomous* real-robot evaluations (see Figure 1). To use AutoEval, human users queue policies for evaluation, which subsequently get evaluated with *minimal* human intervention by the AutoEval system that automatically runs the policy, evaluates the results, resets the scene, and finally returns a detailed evaluation report to the user. AutoEval represents a new paradigm of real-world robot evaluation that has much higher throughput thanks to its minimal reliance on human intervention, allowing for much lower variance results with more trials per evaluation.

There are multiple challenges in designing an effective system for autonomous evaluation of real robot manipulation policies, such as the need for autonomous scene resets and success detection. Our work leverages large pre-trained models to *learn* automatic reset policies and success detectors. Importantly, we adapt these models to the evaluation scene and task at hand to achieve high reliability and minimize the need for human intervention. We propose a general scheme for building automated robot evaluations and instantiate it for common tasks in the popular BridgeV2 robot evaluation environment [77].

Our central contribution is the development of an autonomous evaluation system, AutoEval, that can evaluate user-supplied policies in the real world around the clock. We demonstrate that AutoEval can scale to diverse evaluation environments by instantiating it in three automated evaluation environments for table-top manipulation tasks in the BridgeData V2 environment [77]. Our experiments show that the two aspects of evaluation that typically rely most on human effort, scene resets and success determination, can both be automated with high fidelity, yielding evaluation results that correlate well with ground truth human evaluations. AutoEval drastically increases the evaluation throughput, enabling 500 evaluation episodes per 24-hour period. We also find that AutoEval provides a more reliable policy performance estimate than prior simulated evaluation approaches or offline metrics, while at the same time supporting a wider range of hard-to-simulate tasks like cloth manipulation.

We open-source our code [1] and a detailed step-by-step guide for setting up new AutoEval platforms. Additionally, we open access to multiple Bridge-AutoEval cells, enabling researchers from other institutions to evaluate their policies on our Bridge-AutoEval systems. We hope that this takes a step towards democratizing robotics research and enabling fair comparisons of robot policies on unified evaluation setups.

2. Related Work

Generalist robot policies. There has been significant progress in robot foundation models recently [13, 36, 60, 34, 24, 9, 50, 3, 69, 81, 11, 63], fueled by large-scale robot datasets [16, 77, 35, 68]. These models are trained to perform diverse tasks (e.g., pick-and-place, cloth folding) [77, 36, 11, 63], adapt to various scenes with different backgrounds and distractors [87, 25], and control multiple robot embodiments (e.g., quadrupeds, manipulator arms, drones) [80, 21]. With the increase in capabilities of these generalist robot policies, evaluation becomes ever more time-consuming, because measuring model performance needs evaluations of a variety of different skills and scenes. For example, reporting results for Kim et al. [36] required a few thousand evaluation trials and more than 100 hours of human labor. Evaluation trials needed during development probably compounded this number several times. This makes development and comprehensive evaluation of generalist robot policies increasingly challenging, calling for an

[1] https://github.com/zhouzypaul/auto_eval

evaluation method that is much more scalable.

Robot policy evaluation in the real world. Evaluating robot policies in a fair, comprehensive, and reproducible way is challenging. Robotic methods and systems today are mostly tested in custom settings at the institution where the method is developed. Cross-institution evaluation encounters difficulties with different hardware, task definitions, and performance measures [76]. To address this, multiple works have proposed real robot setups that have reproducible components (such as 3-D printed objects or cheap robot hardware) that are meant to be replicated across institutions [79, 77, 29, 51, 14, 75, 43]. In addition to robot manipulators, there have also been efforts for standardized hardware in other robot embodiments [61, 64]. However, the sensitivity of policies to environmental factors like lighting, camera angles, and robot type makes it hard to accurately reproduce real robot setups across institutions, even when the same set of objects and hardware are used. Others have built evaluation systems that are hosted at a central location to compare different approaches. Some take the form of live competitions [39, 17, 37, 76, 22], while others are hosted at research institutions and open to the public [86, 82]. However, these evaluations all require human involvement to supervise the policy evaluation or to reset the scene, making it expensive in terms of human time and therefore significantly limiting the number of real robot evaluations benchmark participants can perform. In addition, the live competitions are logistically challenging and hard to operate continually. These reproducibility and scalability constraints become even more apparent as the capabilities of robot policies expand to more scenes, tasks, and embodiments. Our approach, AutoEval, can substantially improve the throughput of real robot evaluations by replacing parts of the evaluation pipeline traditionally completed by humans with specialized learned components, thus enabling robots to "evaluate themselves" 24/7. Notably, Bauer et al. [5] proposed a setup for remote, autonomous policy evaluation in the real world as part of their Real Robot Competition, but they focused on evaluations in a single environment, engineered to require no resets and allow for scoring with task-specific, hand-defined rules. In contrast, our AutoEval system is designed for evaluation of *generalist* policies by enabling autonomous evaluation on a wider range of tasks (e.g., pick-place, articulate object & cloth manipulation) via learned reset and scoring modules. While our goal is *not* to build a comprehensive benchmark for robot foundation models, which requires evaluations spanning many tasks, scenes, and embodiments, we demonstrate that our system can be used to automate evaluations for a diverse set of tasks and provide a step-by-step guide to set up new automated evaluation within hours. We hope that by reproducing this recipe at other institutions, the robotics community will over time be able to construct a comprehensive evaluation benchmark for generalist policies.

Evaluation in simulation. While human-run evaluations in the real world are the gold standard used by most prior works, they require extensive human effort and do not scale well as the capabilities of models increase. As a result, *simulation* has been a popular tool for high-throughput evaluation in robot learning research [71, 32, 42, 49, 58, 55, 52, 70, 38, 66, 83, 1, 45, 44, 53, 54]. However, there are still discrepancies between these simulators and the real world, making simulated evaluation different from real-world evaluation. First of all, real-world physics of contacts, collisions, and friction are hard to simulate accurately [74, 33, 18, 41, 59, 78, 4]. Even if the physics simulation is perfect, not all physical parameters can be precisely measured in the real world to replicate in simulation (for example, friction coefficients and actuation delays) [30, 73]. Policies that interact with real-world objects usually exhibit different behavior than they do on their simulated counterparts. Secondly, policies need to deal with real world factors such as noisy and delayed sensory inputs that do not play a big part in simulation. Finally, the visual difference such as texture and lighting between simulated images and real-world observations makes the two types of evaluation quite different [20, 85]. Recent works have tried to reduce the visual discrepancy by building realistic simulators for policy evaluation [47, 46]. SIMPLER [46] constructs high-fidelity replicas of real robot evaluation scenes and demonstrates strong correlation of simulated rollouts to human-run rollouts in the corresponding real robot environments. However, gaps between simulation and the real world remain, and our experiments show that they can affect different policies to varying degrees, leading to inconsistent policy performance rankings between simulation and real world evaluation. Additionally, a large number of tasks, like cloth or liquid manipulation, are challenging to simulate at sufficient fidelity to enable reliable evaluation. In contrast, our approach performs evaluations on real robot systems and thus provides a more reliable signal for policy performance, including on tasks that are hard to simulate, while retaining scalability by minimizing the need for human intervention.

Algorithm 1 Autonomous Policy Evaluation Loop

Input: Task T, policy π to be evaluated, initial state distribution $\rho(s)$, success classifier C_T, reset policy π_T, reset classifier $C_{\rho(s)}$
Output: Estimated prob. of success for task T
for each trial **do**
    Start State: Start from initial state $s_0 \sim \rho(s)$
    Policy Rollout: Rollout π for K steps
    Success Check: Label success using $C_T(s_K)$
    Reset Scene: Rollout reset policy π_T to return initial state to $\rho(s)$
    Failure: If unable to reset or robot unhealthy, notify human operator to help
end for

Autonomous robot operations. Multiple prior works identified the need for human supervision as a key limiting factor in robot learning [87, 2, 34, 65, 15, 40]. While these works typically focus on autonomous policy *improvement* instead of autonomous policy *evaluation*, they share many challenges around robot resets and success detection. Thus, many of the techniques we employ for learning reset policies and success detectors are inspired by prior work in autonomous robot learning, and even some of the metrics are shared, e.g., measuring the frequency of human intervention [6]. However, to our knowledge, our work is the first to explore the design of a general system for autonomous evaluation of generalist policies. While most robot learning researchers are (painfully) aware of the cost of evaluations, existing efforts toward automating real robot evaluations have been limited to task-specific solutions that often involve instrumenting the environment, e.g., with spring-driven or scripted reset mechanisms [57, 19, 34]. In contrast, we provide a task-agnostic, scalable approach for automating robot evaluations with flexible, learned components based on generalizable and broadly applicable foundation models.

3. Autonomous Evaluation of Robot Policies in the Real World

The policy evaluation problem setting we consider is rather straightforward: given a robot policy $\pi(a|o, l)$ that outputs actions given an observation o and language instruction l, and a task definition $T : \mathrm{S} \to \{0, 1\}$ that maps states to task success, we are interested in estimating the probability that the robot policy π would be successful in completing the task T. The output of the policy evaluation is an evaluation score ranging from 0 to 1, representing the success probability.

During robot evaluations, the policy is typically asked to perform the same task multiple times, while applying randomizations to the initial state of the robot and the environment, to get a statistically significant estimate of the policy's performance under the initial state distribution $\rho(s)$. Conventionally, a human evaluator needs to be present for the full duration of the evaluation, supervising the robot, resetting the scene to a new initial position between trials, and scoring the policy's performance. Each individual trial may just take a few minutes, but for generalist policies that need to be evaluated across many tasks and trials, a comprehensive evaluation of a single checkpoint can quickly take multiple days. Thus, we next discuss our AutoEval system for *autonomous* policy evaluation that aims to minimize the required *human* time for robot evaluation.

We present an overview of our AutoEval system in Algorithm 1. At its core, it follows the same structure as a conventional, human-run evaluation, running multiple trials with intermittent resets and performance scoring. However, AutoEval introduces multiple learned modules that automatically perform the tasks that typically require a *human* evaluator. Namely, AutoEval consists of three key modules: (1) a success classifier, that evaluates a policy's success on a given task, (2) a reset policy, that resets the scene back to a state from the initial state distribution upon completion of a trial, and (3) programmatic safety measures and fault detections that prevent robot damage and call for human intervention when necessary. All three components are implemented via flexible, *learned* models, and can thus be easily adapted to automate the evaluation of a wide range of robot tasks. Next, we provide details on the design and training of each component of our AutoEval system.

Success classifier. The success classifier $C_T : \mathrm{S} \to \{0, 1\}$ serves to approximate the ground truth task-success $T : \mathrm{S} \to \{0, 1\}$ that maps image states to a binary success label. Instead of hand-crafting a task-specific success rule as done in prior work [27, 57], AutoEval *trains* a learned success classifier C_T, a recipe which can be easily applied to a wide range of robot tasks. Concretely, we collect a small set of example images of success and failure states. We use approximately 1000 images, which takes less than 10 minutes to collect by tele-operating the robot and saving the frames in the trajectory. We then fine-tune a pre-trained vision-language model (VLM) for the task of binary success detection. Given a language prompt, e.g., "Is the drawer open? Answer yes or no", and an image observation, the

model is trained to predict whether the task was successfully completed. We use a pre-trained VLM to obtain a classifier that is robust to small perturbations of the environment without needing to collect a large number of example images for fine-tuning. In practice, we use the Paligemma VLM [8] for training the success classifier, but many other open-source VLMs would be suitable. More detailed information is provided in Appendix F.

Reset policy. The reset policy $\pi_T(a|s)$ "undoes" what the evaluation policy π did during the evaluation rollout, returning the scene and robot to a state from the initial state distribution $\rho(s)$. Again, instead of relying on task-specific "hardware resets" like springs or magnets, our aim with AutoEval is to design a system that can be flexibly applied to a range of robot tasks. We thus use a *learned* policy for resetting the scene. As we will show in Section 4, scripted reset policies can also be used in some tasks that have more structure, but learned policies provide a more generic approach that can be applied to a variety of tasks. To learn a reset policy, we manually collect a small set of approximately 100 high-quality demonstrations trajectories that reset the scene from plausible end-states of both successful and failed policy rollouts. In practice, this data collection takes typically less than two hours. We then fine-tune a generalist robot policy with behavioral cloning to act as a reset policy. Starting from a generalist policy checkpoint ensures that the reset policy is more robust, and fewer reset demonstrations are required to obtain reliable resets.

Safety detectors. While success detector and reset policy *in theory* enable autonomous evaluations, *in practice* there are numerous issues and edge cases that can prevent evaluations from proceeding autonomously, like robot hardware failures, damage to scene or robot, or out-of-reach objects. In AutoEval we use multiple measures to prevent or gracefully handle such issues. First, we implement a safety workspace boundary that the robot is constrained to, so policies with poor performance do not damage the robot or the AutoEval scene. Second, we implement programmatic checks of the robot's motor status and reboot motors if they failed e.g., due to a collision of the robot with the environment. We also train a "reset success classifier", similar to the success classifier above, that recognizes if resets were successful and re-runs the reset policy otherwise. In both cases, if multiple restarts or resets are not successful, e.g., because an object dropped from the workspace, we implement an automated notification system that requests manual intervention from an "on-call" human operator. In practice, our experiments show that such manual interventions are very rare for the AutoEval cells we implemented (3 interventions per 24 hours of autonomous evaluation, see Fig. 9).

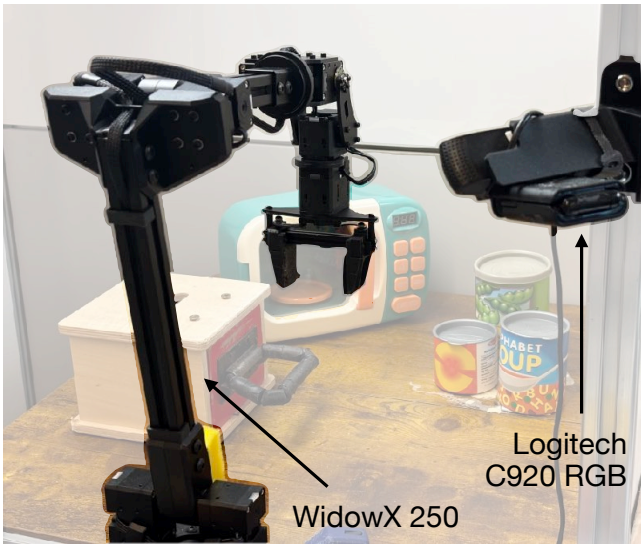

Figure 2: Bridge-AutoEval cell: our robot setup for autonomous policy evaluation in the real world. It consists of a WidowX 250 6-DoF robot arm and a Logitech C920 HD RGB-camera. The scenes reproduce popular evaluation tasks from the BridgeData [77] robot dataset.

Setup time. Overall, we find that the construction of an AutoEval cell for a new task can be completed within 1-3h of human effort, and less than 5 hours total, including model training time for success classifiers and reset policy. This is compared to tens of hours of human evaluation time that can be saved even within a single typical research project. We provide a detailed step-by-step guide for constructing new AutoEval cells in Appendix H to make it easy for others to reproduce AutoEval setups for their own tasks.

4. Bridge-AutoEval: Open-Source Automated Eval Platform

In this section, we describe an instantiation of our automated evaluation system for multiple environments and tasks from the BridgeData V2 dataset [77, 23]. BridgeData is a diverse manipulation dataset containing $60k$+ manipulation demonstrations with a WidowX 6DoF robot arm, that span 13 different skills and 24 environments. State-of-the-art generalist manipulation policies like OpenVLA [36], RT2-X [13], CrossFormer [21], and Pi0 [11, 63] are all trained on BridgeData or a super set of it [16], and therefore policy evaluations on this setup are a natural testbed for scalable evaluation approaches for generalist policies.

Similarly to Walke et al. [77], our Bridge-AutoEval

Figure 3: Three scenes in the Bridge-AutoEval experiments: `sink`, `drawer`, and `cloth`. In total we support five tasks for autonomous evaluation: two pick-and-place task in `sink`, two drawer tasks in `drawer`, and one deformable cloth manipulation task in `cloth`.

setup use a WidowX 250 6-DoF robot arm with a third-person Logitech C920 HD RGB-camera to capture the top-down 256×256 image of the robot workspace, as shown in Fig. 2. We use end effector delta action with blocking control. We built three Bridge-AutoEval cells that can evaluate policies in parallel, as shown in Fig. 3, which we call the `drawer` scene, the `sink` scene, and the `cloth` scene. We maintain constant lighting with an aluminum tripod light over each robot station. Each scene supports evaluation of one to two manipulation tasks: `drawer` supports evaluating "open the drawer" and "close the drawer"; `sink` supports evaluating pick-and-place tasks "put the eggplant in the blue sink" and "put the eggplant in the yellow basket"; `cloth` support the deformable object manipulation task "fold the cloth from top right to bottom left". While none of the exact scenes are in the BridgeData dataset, all scenes are in the distribution of the tasks contained in BridgeData, and have been used in prior works to evaluate generalist policies [36, 87, 84, 10]. We choose these tasks since they represent diverse styles of manipulation tasks: pick-and-place, articulate object manipulation, and deformable object manipulation.

For each scene, we train success classifiers and reset policies following Section 3. We also implement the safety detectors in Section 3 for the WidowX robot (see Appendix A for details), and an automated messaging system to request human interventions by sending push notifications programmatically to a Slack channel with "on call" operators for a given evaluation shift.

One contribution of our work is that we make two of our Bridge-AutoEval cells **publicly available**, so other researchers can schedule evaluations for their policies. We hope that over time, this can contribute to making evaluations in robotics more reproducible and comparable. To make this practical, we provide a public web UI to access our Bridge-AutoEval cells and monitor the evaluation progress, as shown in Fig. 4. Users can choose the task on which they want to perform evaluation, and provide the IP address for a "policy server", that serves the policy they want to evaluate. Given an image observations and a task instruction, the server runs the policy and returns a sequence of 7D actions for the WidowX robot to execute (we provide example code for wrapping user policies in the server interface).

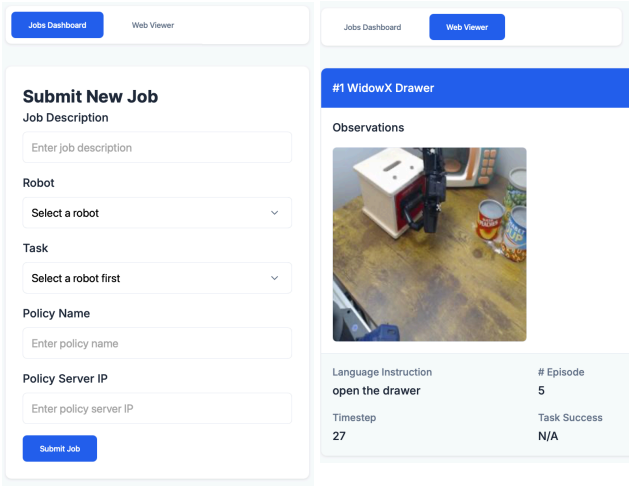

Figure 4: Web UI for submitting evaluation jobs to the Bridge-AutoEval cells. Users choose a desired task and provide the IP address for a policy server they host for evaluation, and can monitor the evaluation through the UI.

Our Bridge-AutoEval system will automatically queue the jobs for evaluation, and query the policy server for robot actions when the policy evaluation is executing. Our AutoEval system can run around the clock, and execute evaluation jobs from all users in the order that they were submitted. At the end of a policy evaluation, AutoEval provides users with downloadable rollout data and a detailed performance report of the autonomous evaluation, which contains rollout videos, success rates, episode durations, and frequencies of motor resets or required human interventions. Fig. 5 shows part of an example report, which is accessible online instantly after AutoEval finishes. A step-by-step guide for submitting your policies to AutoEval can be found at https://auto-eval.github.io.

5. Experimental Results

The goal of our experiments is to answer the following questions: (1) How well does AutoEval's policy performance estimates match those of "oracle" human-run evaluations? (2) Can AutoEval evaluate policies more reliably and on a wider range of tasks than prior approaches for scalable evaluation of generalist policies? (3) How stable is AutoEval in operations over long periods of time and how effectively can AutoEval minimize the amount of required human operator time?

Figure 5: Excerpt from an AutoEval result report, provided to the user upon completion of the automated evaluation. Users can see the per-episode success rate, rate of evaluation progress, instances of automatic recovery from motor failures, and qualitative rollout videos as well as classifier result plotted with initial and final frames to obtain a wholistic understanding of the policy's performance.

5.1. Experimental Setup

Tasks. We evaluate policies on the five Bridge V2 [77] evaluation tasks described in Section 4: opening and closing a drawer, placing a plastic eggplant in a sink and a basket, and folding a piece of cloth. All tasks are performed using a WidowX 6-DoF robot arm. During human-run evaluations, success is counted when the drawer is completely closed or opened at least 1.5cm, respectively, if the eggplant is fully inside the sink or basket at the end of the episode, and if the cloth is folded to at least a quarter of the way diagonally. We randomize the initial position of the eggplant, drawer, and the cloth at the beginning of each episode.

Policies. We run evaluations with six recently released generalist robot policies from the robotics community: **OpenVLA** [36], a 7B parameter vision-language-action model (VLA) pre-trained on the Open X-Embodiment dataset [16], **Octo** [72], a 27M parameter transformer policy, also pre-trained on Open X-Embodiment, **Open-π_0** [67], an open-source reproduction of the 3B parameter π_0 VLA [11] (the original π_0 was not available in open-source at the time of writing), pre-trained on the Bridge V2 dataset, **MiniVLA** [7], a 3B parameter VLA pre-trained on the Bridge V2 dataset [77], **SuSIE** [10], a hierarchical policy that combines a image diffusion subgoal predictor with a small diffusion low-level policy, pre-trained on Bridge V2, and **SuSIE-LL**, which directly executes the goal-conditioned behavioral cloning low-level policy from SuSIE. This set of policies is a representative sample of current state-of-the-art generalist policies. All policies contain the Bridge V2 dataset as part of their training data, and we evaluate the publicly released checkpoints for all models.

Figure 6: SIMPLER [46] simulated evaluation scenes for the tested environments. Simulated evaluation is fast and cheap, but can struggle from visual and physics discrepancies between simulation and the real world.

Comparisons. We compare multiple approaches for scalable evaluation of generalist policies. Concretely, we compare our approach, **AutoEval**, to prior work on simulated evaluation of robot manipulation policies, **SIMPLER** [46]. SIMPLER builds realistic simulated versions of real-world environments and evaluates policies purely in simulation. For our experiments, we reuse the existing SIMPLER environment for the Bridge sink environment, and build a new SIMPLER simulation environment for the drawer scene (Fig. 6) by following Li et al. [46]'s step-by-step guide. Deformable objects such as the cloth in our `cloth` scene are hard to simulate in general [28, 48], and at the time of writing the simulator of SIMPLER, Maniskill [56], does not support simulating deformable objects so we do not evaluate the `cloth` scene in simulation. In addition, we compare to using mean-squared error on a validation set ("**val-MSE**") as a scalable approach for offline evaluation of robot policies.

Metrics. Human-run real-world evaluations represent a gold standard for robotic policy evaluation. For scalable evaluation approaches like the ones we compare in this work, the goal is to approximate the result of such human-run evaluations as closely as possible, while being significantly more scalable to run. Following Li et al. [46] we use two metrics to measure how closely the respective evaluation results match those of human-run evaluations: (1) **Pearson correlation** [62], which measures the linear consistency between two random variables, and is a widely used statistical tool for assessing correlation, with scores nearing 1 indicating high correlation. (2) **MMRV** (Mean Maximum Rank Violation) [46], which measures the consistency of policy

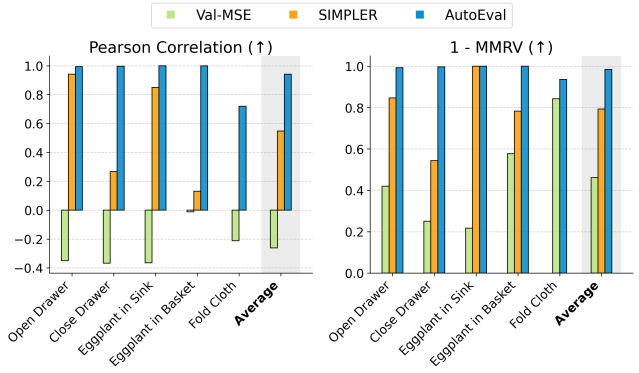

Figure 7: Correlation of scalable evaluation approaches to oracle human-run evaluations. AutoEval closely matches human evaluations, achieving high correlation and low MMRV score (plotted in the figure is $1 - \text{MMRV}$ for clarity). In contrast, SIMPLER simulated evaluations and validation MSE do not correlate as well with human evaluations.

ranking and, as described in Li et al. [46], can be more robust to noise on the evaluation results. MMRV is computed as follows: given N policies $\pi_{1..N}$ and their respective success rates $R_{A,1..N}$, $R_{B,1..N}$ estimated via two evaluation procedures A and B, we compute:

$$\text{RankViolation}(i, j) = |R_{A,i} - R_{A,j}| \cdot \mathbf{1}[(R_{B,i} < R_{B,j}) \neq (R_{A,i} < R_{A,j})]$$

$$\text{MMRV}(R_A, R_B) = \frac{1}{N} \sum_{i=1}^{N} \max_{1 \leq j \leq N} \text{RankViolation(i, j)}.$$

For each evaluation approach we compute MMRV with reference to human-run "oracle" evaluations, where low MMRVs indicate closely matching evaluation results.

5.2. AutoEval Closely Matches Human Evaluation Results

In this section, we test how well the different evaluation approaches from Section 5.1 match results from human-run evaluations. For each evaluation method, we run 50 evaluation rollouts for each policy in each task (except "val-MSE", which does not require rollouts).

We report results in Fig. 7, with a detailed breakdown of results per task, policy, and evaluation method in Appendix, Table 1 to Table 3. Similar to prior work [46], we find that simple validation MSE is a poor evaluation metric for robot policies: it actually negatively correlates with real robot performance and thus does not provide a reliable performance estimate. We find that SIMPLER evaluations in simulation provide a better performance signal, but lack reliability. Concretely, our results show that SIMPLER occasionally matches real-world performance well (e.g., for the "open drawer" task), but in other cases not accurately reflects the policy's performance. For example for Open-π_0 in the "put eggplant to sink" task, the policy performs very poorly in simulated evaluations, but achieves high success rate in the real world. Intuitively, different policies may suffer differently from the remaining sim-to-real gap in SIMPLER evaluations. As a result, SIMPLER's effectiveness is policy dependent and it cannot provide a *reliable* policy evaluation.

In contrast, we find that our approach, AutoEval, closely matches the results of oracle human-run evaluations, with an average Pearson score of 0.942 and MMRV of 0.015 (plotted as $1 - \text{MMRV}$ in Figure 7 of 0.985). In particular, an MMRV score close to zero indicates that it rarely disrupts the ranking of policies. Intuitively, since evaluations are still run in the real world, there is no sim-to-real gap that could negatively affect policy performance. In practice, we find that success detector and reset policy work reliably during evaluation. We show qualitative examples of autonomous evaluation rollouts in Fig. 8, and further examples in Appendix B. Importantly, we find that AutoEval drastically reduces the human effort required to run real robot evaluations, cutting the human evaluator time for robot evaluations by $> 99\%$ compared to conventional, human-run evaluations. We also note that AutoEval does not perfectly match human-run evaluation results, due to occasional failures in success detection and reset policy. However, we find that in practice the accuracy of AutoEval is sufficient to provide a strong ranking signal.

5.3. AutoEval Robustly Runs Over Long Time Spans

A key advantage of autonomous robot evaluations is that they can run 24/7, since they require little human involvement. In this section, we test AutoEval's stability when operating over extended periods of time, both in terms of its up-time and the reproducibility of policy evaluation performance.

For this investigation, we performed a long-running evaluation over the course of 24 hours, repeatedly interleaving the evaluation of various policy checkpoints, using the "*open drawer*" and "*close drawer*" tasks. In Fig. 9, we present the evaluation throughput, as well as the number of human interventions needed over the span of the whole 24 hours. We present evaluation throughput in terms of the number of valid evaluation steps taken per minute (excluding reset policy steps and re-evaluation steps needed because of motor failure). Over the course of a day, a single AutoEval cell is able to run $60,000$ evaluation steps (roughly 850 episodes on the

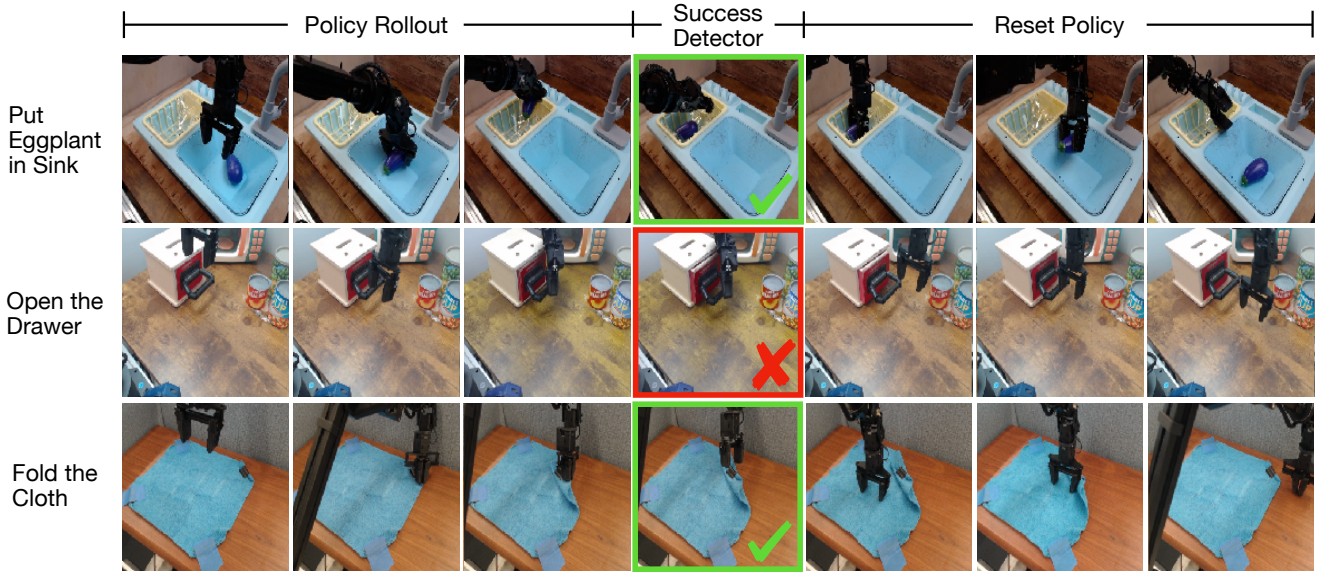

Figure 8: Qualitative visualization of AutoEval evaluation rollouts on three of our tasks. After the policy execution is done, the success classifier determines whether the rollout was successful. Then, the reset policy returns the environment into a state from the initial state distribution for the next evaluation. Our evaluations cover representative robot manipulation tasks: pick-place, articulate and deformable object manipulation.

drawer scene), with an average speed of 42 evaluation steps per minute. The AutoEval throughput varies in Fig. 9 because of the different inference speed of different policies. The average AutoEval speed, shown in dotted blue line, is slightly lower but on par with the average evaluation speed of a human evaluator performing manual resets of the environment and recording success rates. Even though AutoEval has a slightly lower throughput, AutoEval runs autonomously and only required a total of three human interventions in the span of 24 hours to reset the scene or robot. Every time a human operator needed to intervene, they simply needed to check and reset the objects' position in the scene, and potentially move the robot arm into reset position if a motors failed and the robot fell on the table. Afterward, the human operator can make AutoEval resume simply with the press of a button. Assuming the average human response time during the day is 30 minutes and 8 hours at night, and that the 3 required resets occur randomly throughout the 24 hours, a whole day of AutoEval yields $\approx$ 19 hours of "real evaluation time" that is not blocked by human reset. Assuming that each human reset operation takes 1 minute, 19h of real autonomous evaluation only costs 3 minutes of human time, compared to $\approx$ 16 hours if a human evaluator wanted to run the same number of trials by hand. This means that AutoEval can reduce human time involvement by **>99%**.

Are AutoEval results consistent across time? We test the *consistency* of AutoEval evaluations, i.e., AutoEval's ability to produce comparable performance estimates across multiple iterations of evaluating the same policy. To test this, we run the Open-π_0 policy through a sequence of 9 evaluations on the "open drawer" task, each consisting of 50 individual trials, or a total of 450 trials. Using AutoEval, the full evaluation takes ~11 hours. We report the results of this evaluation in Fig. 10. We find that AutoEval produces consistent evaluation results across long periods of time. Concretely, for the first 7 evaluation runs, or a total of 350 evaluation episodes, AutoEval performance evaluation are within the margins of what might be considered the natural variance of robot evaluations ($\pm$10%). We see a regression in performance after approximately 8 hours of continuous operation, which we attribute to an overheating of the motors of our rather affordable WidowX robot (<$3500) after many hours of operation. To mitigate the effects of such overheating in practice, we pause autonomous evaluations for 20 minutes every 6 hours to let the motors cool off before resuming evaluations.

In addition, we evaluate AutoEval's performance over two months of continuous operation. Results in Appendix J shows the AutoEval yield consistent results over such long time periods.

5.4. Analyzing AutoEval Failure Modes

While our previous experiments show that AutoEval closely matches human-run evaluations, we observe that over extended periods of operation errors occur occasionally. To better understand the sources of these errors

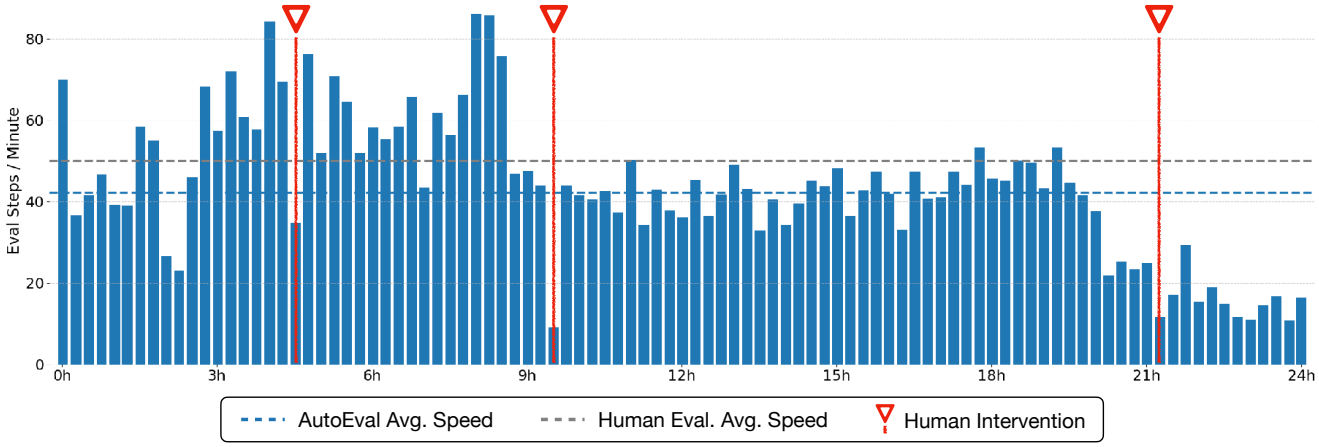

Figure 9: Visualization of a 24 hour AutoEval evaluation run with ~850 total evaluation episodes. AutoEval ran autonomously over extended periods of time and only required a total of 3 human interventions over a 24 hour period. On average, the evaluation throughput of AutoEval is on par with that of human evaluations, but saves **99%+** human operator time.

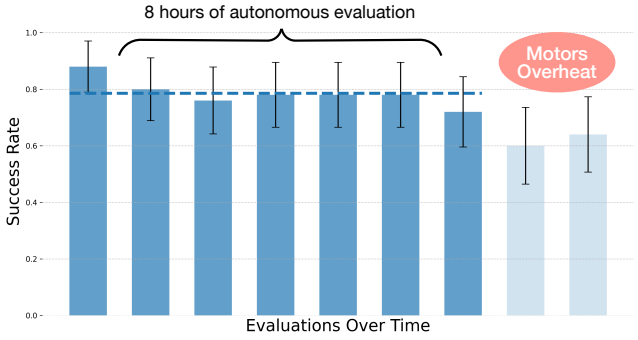

Figure 10: AutoEval evaluation scores remain consistent over 8 hours of autonomous evaluations. After 8 hours, WidowX motors overheat and evaluation scores start to drift. As a result, we pause evaluation for 20 min every 6 hours to let the motors cool off. Error bars show 95% confidence intervals.

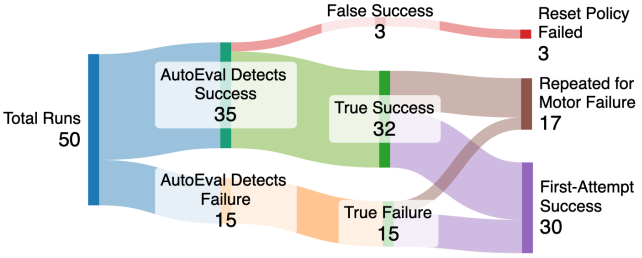

Figure 11: Analyzing 50 AutoEval runs on the `sink` scene: the main failure modes is false positive results because the reset policy failed to reset the scene.

and help the design of future autonomous evaluation cells, we perform a detailed analysis of all failures occurring in a 50 episodes AutoEval run on the "put eggplant in blue sink" task with the Open-π_0 policy. We visualize the outcome of our analysis in Fig. 11. While many episodes experienced motor failure because of harsh contact with the scene, AutoEval handles such failure automatically by re-running those trials, and only report evaluation trials that do not contain motor failures. We find that for only three out of 50 trials, the autonomous evaluation fails, since the episodes mistakenly get classified as successes and the reset policy fails.

One key takeaway from this failure analysis is that our Bridge-AutoEval setup is already very reliable with few errors, and that most room for improvement is in improving the *efficiency* by reducing the number of motor failures during evaluation, e.g. by implementing a more compliant robot controller that prevents harsh environment interactions.

6. Conclusion

In this work, we introduced AutoEval, a system for autonomous evaluation of generalist robot policies in the real world. We demonstrated that AutoEval can perform high-quality evaluations around the clock and with minimal human involvements across a range of commonly used robot evaluation tasks. Our experiments shows that AutoEval evaluation results closely matches those of human-run evaluations, and are both more reliable and applicable to a wider range of tasks than prior simulation-based evaluation approaches. In an effort to make real-robot evaluation widely available and more comparable, we provide public access to two AutoEval evaluation cells for popular BridgeData V2 evaluation tasks, for which users can submit their policies online for evaluation, and receive detailed evaluation reports. We hope that this work will inspire more AutoEval evaluation cells to be set up across institutions to form a diverse automated evaluation framework, which will significantly speed up robot learning research.

7. Limitations

AutoEval environment creation time. Our current approach for creating new environments for automated evaluation requires some up-front human effort to train the reset policy and success classifier. In our experience, the complete process only takes a few hours for a new scene and is quickly outweighed by the time savings of autonomous evaluation, but future work can explore more efficient ways of constructing reset policies and success classifiers to further reduce the effort for setting up a new scene for autonomous evaluation. We also expect that future improvements to vision foundation models and generalist policies will make the training of robust success classifiers and reset policies easier, possibly to the point where we can "train" these modules simply by providing a handful of examples in context.

Evaluating policy robustness. There are various dimensions of out-of-distribution robustness we may be interested in evaluating for robot policies, e.g., robustness to varying camera angles, distractor objects, lighting conditions, or table textures (see Gao et al. [26] for a more comprehensive taxonomy). Varying each of these axes in a controlled way as part of an *automated* evaluation pipeline may require major engineering efforts, and AutoEval currently does not support such evaluations. In the future, a decentralized network of AutoEval cells may be able to increase evaluation diversity across many of these axes.

Mobile manipulation tasks. Our experiments capture a set of robot manipulation tasks that are reflective of the kind of tasks commonly used for evaluating generalist robot policies today, where the primary focus is on tabletop manipulation tasks. We believe that our approach will transfer well to a wide range of other single-arm and bi-manual table top manipulation tasks. However, mobile robot tasks, particularly mobile manipulation tasks, may pose new challenges e.g. with regards to robust resets at room scale, success estimation under partial observability, and operational safety, all of which pose important directions for future work.

Binary success metrics. AutoEval evaluation currently only supports binary success estimates (did the policy succeed at a task or fail). When humans run evaluations, they can provide a more granular assessment of the policy's performance, including task progress scores and a qualitative analysis of the policy's proficiency. While AutoEval users can obtain similar assessments from re-watching the logged evaluation videos, this is a time-consuming process. In future work, it would be exciting to investigate whether more granular performance analysis can be provided in an automatic evaluation framework, e.g., by querying powerful video summarization models.

Acknowledgments

We would like to thank Kyle Stachowicz and Mitsuhiko Nakamoto for valuable discussions and feedback on an earlier version of the system. This work was partly supported by ONR N00014-25-1-2060 and NSF IIS-2150826. Pranav is supported by the NSF Graduate Research Fellowship.

Author Contributions

Zhiyuan designed and implemented the AutoEval system and Web UI and led the experiments. Pranav assisted in setting up the AutoEval stations and running experiments. You Liang implemented the comparison experiments with SIMPLER and baseline methods and contributed to the AutoEval station setup. Together, all three iterated on reset policy and success detector training. Karl and Sergey provided valuable guidance throughout the project and contributed to the paper writing.

References

[1] Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

[2] Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Edward Lee, Sergey Levine, Yao Lu, Sharath Maddineni, Kanishka Rao, Dorsa Sadigh, Pannag Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Steve Xu, and Zhuo Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

[3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[4] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

[5] Stefan Bauer, Manuel Wüthrich, Felix Widmaier, Annika Buchholz, Sebastian Stark, Anirudh Goyal, Thomas Steinbrenner, Joel Akpo, Shruti Joshi, Vincent Berenz, et al. Real robot challenge: A robotics competition in the cloud. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 190–204. PMLR, 2022.

[6] Jenay M Beer, Arthur D Fisk, and Wendy A Rogers. Toward a framework for levels of robot autonomy in human-robot interaction. *Journal of human-robot interaction*, 3(2):74, 2014.

[7] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL https://github.com/Stanford-ILIAD/openvla-mini.

[8] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[9] H Bharadhwaj, J Vakil, M Sharma, A Gupta, S Tulsiani, and V Kumar. Roboagent: Towards sample efficient robot manipulation with semantic augmentations and action chunking, 2023.

[10] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[11] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[12] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[13] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[14] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

[15] Annie S Chen, HyunJi Nam, Suraj Nair, and Chelsea Finn. Batch exploration with examples for scalable robotic reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4401–4408, 2021.

[16] Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin

Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[17] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[18] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 1. 2015.

[19] David B D'Ambrosio, Saminda Abeyruwan, Laura Graesser, Atil Iscen, Heni Ben Amor, Alex Bewley, Barney J Reed, Krista Reymann, Leila Takayama, Yuval Tassa, et al. Achieving human level competitive robot table tennis. *arXiv preprint arXiv:2408.03906*, 2024.

[20] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020.

[21] Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[22] Earth Rover Challenge Team. The Earth Rover Challenge, 2025. URL https://sites.google.com/view/the-earth-rover-challenge. Accessed: 2025-02-01.

[23] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[24] Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, et al. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv preprint arXiv:2312.02976*, 2023.

[25] Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

[26] Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot policies. *arXiv preprint arXiv:2503.01238*, 2025.

[27] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[28] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022.

[29] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-

world benchmark for long-horizon complex manipulation. *arXiv preprint arXiv:2305.12821*, 2023.

[30] Peide Huang, Xilun Zhang, Ziang Cao, Shiqi Liu, Mengdi Xu, Wenhao Ding, Jonathan Francis, Bingqing Chen, and Ding Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In *Conference on Robot Learning*, pages 734–760. PMLR, 2023.

[31] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[32] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[33] Arthur Juliani. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

[34] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[35] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[36] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[37] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347, 1997.

[38] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[39] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. The darpa robotics challenge finals: Results and perspectives. *The DARPA robotics challenge finals: Humanoid robots to the rescue*, pages 1–26, 2018.

[40] Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, et al. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. *arXiv preprint arXiv:2312.11374*, 2023.

[41] Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C Karen Liu. Dart: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500, 2018.

[42] Youngwoon Lee, Edward S Hu, and Joseph J Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://clvrai.com/furniture.

[43] Jürgen Leitner, Adam W Tow, Niko Sünderhauf, Jake E Dean, Joseph W Durham, Matthew Cooper, Markus Eich, Christopher Lehnert, Ruben Mangels, Christopher McCool, et al. The acrv picking benchmark: A robotic shelf picking benchmark to foster reproducible research. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 4705–4712. IEEE, 2017.

[44] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[45] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai,

Benjamin Martinez, et al. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024.

[46] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[47] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, et al. Openrooms: An open framework for photorealistic indoor scene datasets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7190–7199, 2021.

[48] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[49] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[50] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[51] Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, page 02783649241276017, 2023.

[52] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[53] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[54] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

[55] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[56] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[57] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[58] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[59] NVIDIA. Physx, 2020. URL https://developer.nvidia.com/physx-sdk.

[60] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[61] Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang,

et al. Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504. IEEE, 2017.

[62] Karl Pearson. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242, 1895.

[63] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[64] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017.

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

[66] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.

[67] Allen Z. Ren. Open-pi-zero: An open-source hardware project. https://github.com/allenzren/open-pi-zero, 2024. Accessed: 2025-01-31.

[68] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[69] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[70] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

[71] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[72] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[73] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[74] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[75] Stephen Tyree, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Jeffrey Smith, and Stan Birchfield. 6-dof pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13081–13088. IEEE, 2022.

[76] Karl Van Wyk, Joe Falco, and Elena Messina. Robotic grasping and manipulation competition: Future tasks to support the development of assembly robotics. In *Robotic Grasping and Manipulation: First Robotic Grasping and Manipulation Challenge, RGMC 2016, Held in Conjunction with IROS 2016, Daejeon, South Korea, October 10–12, 2016, Revised Papers 1*, pages 190–200. Springer, 2018.

[77] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-

Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[78] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

[79] Brian Yang, Jesse Zhang, Dinesh Jayaraman, and Sergey Levine. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *ICRA*, 2019.

[80] Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.

[81] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

[82] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

[83] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[84] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[85] Jingwei Zhang, Lei Tai, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4(2):1148–1155, 2019.

[86] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, et al. Train offline, test online: A real robot learning benchmark. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9197–9203. IEEE, 2023.

[87] Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

A. Safety During Extended Autonomous Robot Operations

To ensure that the robots can autonomously and safely operate for a long time, we take several measures to ensure the safety of the robot and to preserve the scene. First, we set safety boundaries for the robot such that the policy cannot go beyond certain xyz axis (e.g. beyond the view of the camera) so that it does not run into objects unintentionally. Second, since the WidowX robot arms do not natively support impedance control, we limit the maximum effort on each of the robot joints, so that ineffective policies do not press too hard against objects and cause motor failure or damage the scene. The common robot failure is due to joint failure when interacting and colliding with the objects in the scene, hence we constantly monitor and log the joint effort values, software reboot the joints at a safe arm position when joint errors are detected during each trial. Third, we use the safety detectors described in Section 3 to monitor and out-of-distribution and unexpected scenarios. Finally, we further ensure safety of the scene by taping the drawer and cloth to the table to prevent them from falling off the table, and add a thin plastic wrap over the yellow sink to prevent robot gripper getting jammed and damaged.

B. Visualizations of AutoEval Rollouts

Figure 12 presents evaluation trajectories in the five different Bridge-AutoEval tasks. The actual language commands fed to the evaluated policies are:

1. *"Close the drawer"*
2. *"Open the drawer"*
3. *"Put the eggplant in the yellow basket"*
4. *"Put the eggplant in the blue sink"*
5. *"fold the cloth from top right to bottom left"*

C. Detailed Evaluation Results on Bridge-AutoEval

In Table 1 to Table 2, we provide detailed evaluation results for our comparison of different scalable evaluation approaches across the five Bridge V2 evaluation tasks. For both tasks on the `Drawer` scene, we evaluate all policies for 70 steps at maximum; for both tasks on the `Sink` scene, we run 100 steps; for the `Cloth` scene, we run 80 steps.

Policy	Drawer		Sink		
	Open Drawer	Close Drawer	To Basket	To Sink	Fold Cloth
OpenVLA	40/50	46/50	1/50	0/50	13/50
Open π_0	29/50	46/50	7/50	47/50	12/50
Octo	1/50	5/50	0/50	0/50	4/50
SuSIE-LL	0/50	1/50	0/50	0/50	0/50
SuSIE	1/50	18/50	0/50	0/50	9/50
MiniVLA	33/50	49/50	38/50	0/50	8/50

Table 1: AutoEval results on five Bridge-AutoEval tasks across six different generalist policies.

Policy	Drawer		Sink		
	Open Drawer	Close Drawer	To Basket	To Sink	Fold Cloth
OpenVLA	40/50	46/50	1/50	0/50	12/50
Open π_0	24/50	45/50	7/50	47/50	3/50
Octo	0/50	0/50	0/50	0/50	2/50
SuSIE-LL	0/50	0/50	0/50	0/50	0/50
SuSIE	2/50	13/50	0/50	0/50	10/50
MiniVLA	32/50	49/50	38/50	0/50	8/50

Table 2: Ground truth human evaluation results for the five Bridge-AutoEval tasks across six different generalist policies.

D. Evaluation on Bridge-SIMPLER [46]

In AutoEval , we introduced a new Drawer Scene to the existing SIMPLER [46] setup for the WidowX robot. The scene was visually matched with the AutoEval's Drawer setup, and overlaid with the same background to ensure consistency. A 3D model of the drawer, with exact dimensions matching the real-world setup, was also created. This scene introduced two evaluation tasks: *"open and close the drawer"*. To add variability to each evaluation trial, we randomized the end effector's initial pose, the drawer's initial pose, and the lighting conditions in the background.

In addition to the Drawer Scene, AutoEval includes a Sink Setup, which closely resembles the existing SIMPLER [46] Sink Scene. In SIMPLER, the task here is the *"move the eggplant to the basket"* task. We also introduced a reverse task, *"move eggplant to the sink,"* effectively making the scene reset-free. This allows for both forward and reverse tasks in the same environment.

With these two scenes and four tasks, we conducted 50 runs for each scene across five different generalist policies. The detailed results are shown in Table 3

Policy	Drawer Scene		Sink Scene	
	Open Drawer	Close Drawer	To Basket	To Sink
OpenVLA	32/50	2/50	1/50	0/50
Open π_0	34/50	24/50	45/50	6/50
Octo	3/50	0/50	6/50	3/50
SuSIE-LL	1/50	0/50	0/50	0/50
SUSIE	0/50	41/50	7/50	0/50
MiniVLA	30/50	23/50	10/50	2/50

Table 3: Evaluation Results on SIMPLER [46] for Drawer and Sink Scene on four tasks and six different policies.

E. Computing Action Validation MSE between policies

We sample 400 trajectories from the validation set of BridgeData [77] to compute the action mean squared error (MSE) for each policy. The results are shown in Table 4. Consistent with the findings in SIMPLER [46], this illustrates a weak correlation of task success rate on AutoEval with validation MSE.

Policy	200 Trajectories		400 Trajectories	
	MSE	Norm MSE	MSE	Norm MSE
OpenVLA	0.0143	1.362	0.015	1.431
Open π_0	0.082	1.433	0.085	1.495
Octo	0.0214	1.504	0.023	1.611
GCBC	0.008	0.817	0.009	0.870
SUSIE	0.018	1.1579	0.018	1.244

Table 4: Average Validation MSE across Policies on 400 random trajectories from BridgeV2 Dataset. Norm MSE represents the MSE of normalized actions, while MSE represents the MSE of raw action magnitudes.

Figure 12: Samples of autonomous policy evaluation trials with AutoEval on the five tasks. The classifier result of each task is visualized on the right hand side, and the reset policy is not shown.

F. Success Classifier in Bridge-AutoEval cells

To train success classifiers for the Bridge-AutoEvalscenes, we finetune the Paligemma VLM to act as a classifier. We manually collect a dataset of roughly 1000 images for each scene, and manually label them. We form VQA questions with the labels, and finetune the base 3B parameter VLM with quantized LoRA using a learning rate of $2e-5$, batch size of 4 for 80 iterations. For some scenes, we combines the success classifier and the safety detector into a single fine-tuned VLM: we train the VLM to output "invalid" (in addition to classifying success) when there are out-of-distribution cases that prevent evaluation from proceeding autonomously (e.g., object out of reach).

For each evaluation scene, approximately 1000 image frames are collected to fine-tune the VLM. The corresponding language prompts are:

1. Sink Scene: *"is the eggplant in the sink or in the basket? answer sink or basket or invalid"*
2. Drawer Scene: *"is the drawer open? answer yes or no"*
3. Cloth Tabletop Scene: *"is the blue cloth folded or unfolded? answer yes or no"*

We evaluate our classifier both by running it on a held-out test set of roughly 100 images and by teleoperating the robot and running the classifier on all the image observations throughout the trajectory. We choose to deploy success classifiers in AutoEval that have an accuracy of $> 95\%$. When the classifier trained on the initial ~ 1000 images does not achieve this accuracy threshold, we found it helpful to improve classifier performance by rolling out the trained model, identifying incorrect predictions and collecting these images, and retraining on these "hard" examples.

G. Reset Policy in Bridge-AutoEval cells

To train reset policies for the Bridge-AutoEvalcells, we finetune the generalist OpenVLA policy with LoRA, with batch size 64 and learning rate 10^{-4} for 1000 iterations. For each scene, we collect $50-100$ demonstration trajectories via teleoperation, and train with a standard behavior cloning loss. We also use a scripted policy for one of our reset policy - "Close the Drawer" task, where the reset success rate is not sensitive to variation in scene.

Similar to the success classifiers, we choose to deploy reset policies that have a success rate of $> 95\%$.

H. Step-by-step AutoEval Construction Guide

Below, we provide a step-by-step guide for creating an AutoEval setup for a new evaluation tasks. Refer to our code release at https://github.com/zhouzypaul/auto_eval for code on each of the steps and detailed instructions on how to run the code. The full process takes approximately 3 hours of active human effort and a total of 5 hours including model training time for reset policy and success detector.

1. **Train Reset Policy**: Start by collecting approximately $50 - 100$ high-quality robot demonstrations of resetting behavior from sensible final states of policy rollouts. Try to cover a diverse set of "reset start states", including those that failed the original task, to obtain a robust reset policy. Once you collected the dataset, fine-tune a generalist policy like OpenVLA [36], e.g., using LoRA fine-tuning, on your the small demonstration dataset. If you find that the reset is unreliable and fails often, consider collecting more reset demonstrations particularly on the positions where the reset policy fails and re-train the policy. For a small set of tasks that has more structure, you can also use *scripted* policies to reset the scene. See our code release for code to record tele-operated policies for WidowX robots and replaying it to reset the scene. An easy way to make reset policies stronger is to simply execute it for multiple times if it fails. Proceed when your reset has success rate of $> 95\%$.
2. **Train Task Success Classifier (And Safety Detector)**: While the success classifier and the safety detectors serve two different functions, in practice you can train a combined three-way (success, failure, invalid) classifier that acts as both the success and safety detector. This classifier will output "invalid" when OOD events happen (e.g. objects out of reach) and human intervention is needed, else it will output whether the task is successfully completed or not. Collect approximately 1000 images of success and failure (and invalid) states. Be sure to collect lots of failures (and invalid) states because there are many ways in which the robot can fail. Then fine-tune a vision-language model like Paligemma [8] on this dataset. Test the performance of your classifier by tele-operating the robot and scoring the observations along the trajectory. You can improve the classifier by saving the observations that it mis-classifies and re-train by incorporating these "hard examples" into the original dataset of images. Proceed when your success classifier has accuracy $> 95\%$.
3. **Set Up Safety and Robustness Measures**: We have implemented multiple safety measures described in Appendix A for the WidowX robot. To set up new AutoEval tasks on WidowX robots (or ViperX or similar robots), you can directly use our infrastructure; to set up AutoEval on a new robot embodiment, consider implementing the following safety measures. First, if your robot does not have an integrated p-stop that prevents forceful collisions with the environment, implement a limit on the motor current to prevent high-force contact with the environment that may damage the robot and the scene. Also implement a software mechanism to reboot the motors when they fail. Second, use a workspace boundary to limit the reach of the robot: limit the robot from reaching out-of-scene objects and prevent the robot from removing objects that are in the scene. Finally, use an "on-call" system that sends push notifications to human monitors when the robot reports an irrecoverable safety issue or the reset policy fails for $N \approx 3$ times in a row (as determined by the success detector). We implement a Slack bot that sends notifications through Slack channels.
4. **Prepare for Policy Submission**: Power on the robots and start the low-level robot controllers. We set up the robot environment as a server that waits to receive actions from the policy. Then, start the webserver to access the UI for tracking the evaluation jobs queue and submitting a policy through the webapp. Next, host your policy that needs to be evaluated as a server. Finally, submit the IP and port of your policy server to the AutoEval web UI as an evaluation job to the AutoEval system. The evaluation job will be automatically queued and ran. See the code release for more detailed instructions.

I. Bridge-AutoEval Deployment Details

As described in Section 4, we open access to our Bridge-AutoEval cells to the research community. The two different AutoEval cells accepts and executes jobs in parallel. While the two WidowX robots will accept evaluation jobs 24/7, we enforce a 20 minute rest period every 6 hours where the robot will torque off and let the motors cool off (see Fig. 10 for why this is necessary). The reset period will only happen between evaluation

jobs.

Since we host the reset policies for the four tasks in Bridge-AutoEval 24/7, we optimize for lightweight policies (as compared to the fine-tuned OpenVLA reset policy we use in Section 5). For the two tasks on `Drawer`, we use scripted reset poliy; for the two tasks on `Sink`, we fine-tune MiniVLA [7] on the same demos. We find that all reset policies have success rate $> 95\%$.

J. Evaluation Results Reproducible Across Months

We find that AutoEval reproduces results even after more than 2 months of continued use, demonstrating its robustness to aging effects. We compare AutoEval results that are two months apart for two policies on three tasks as shown in Table 5. During the two months, AutoEval operated continuously for a rough total of 200 hours. Table 5 shows that all evaluations perform similarly when evaluated two months apart, and the reset policy and success classifiers still have accuracies **96%** and **96%** respectively. We attribute such robustness to (1) safety controllers (Appendix A) limiting robot joint efforts to prevent high-force contact and damages, and (2) foundation model pre-training (Paligemma VLM, OpenVLA) making policies and detectors resilient to minor scene changes. Over two months, we have observed minimal "aging" – e.g. there are light scratches on the drawer upon close examination, but they are not visible in the 256x256 pixel policy image observations and does not impact the drawer physics.

Policy	Open Drawer	Close Drawer	Eggplant to Basket
OpenVLA (old)	39/50	48/50	4/50
OpenVLA	40/50	46/50	1/50
Avg. Δ Success	+2%	-2%	-6%
Open π_0 (old)	30/50	45/50	9/50
Open π_0	29/50	46/50	7/50
Avg. Δ Success	-2%	+2%	-4%

Table 5: AutoEval results obtained two months apart: results remain highly consistent across two different policies on three different tasks. All correlate well to human evaluations.

K. Initial States in Bridge-AutoEval Cells

We find that our learned reset policy is able to reset to a consistent distribution of initial states. As an example, we plot the centroids of all eggplant initial positions for three representative AutoEval runs of the "Eggplant to Basket" task in Fig. 13 (50 trials each). Qualitatively, we find that the reset distributions of other tasks are similarly overlapping, and also roughly cover the task distribution.

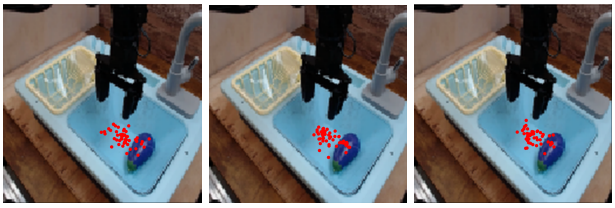

Figure 13: Initial state distribution for 3 different AutoEval runs is consistent. Red dots show the centroid position of the eggplant. Each run uses 50 trials.

L. Improving AutoEval with Additional Human Involvement

Though AutoEval results highly correlate with ground truth human-run evaluations, it is not perfect (as shown in Fig. 11). Additional human effort, when available, can further improve AutoEval's accuracy. The easiest and most effective way to apply extra human effort can be spent going through the evaluation report after AutoEval finishes to remove the runs where the reset policy fails, and relabel the success manually. Going through 50 trials of AutoEval roughly takes $1-2$ minutes of human time. This enables ground-truth judgment of evaluations runs while still saving the majority of human time required in robot evaluations.