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

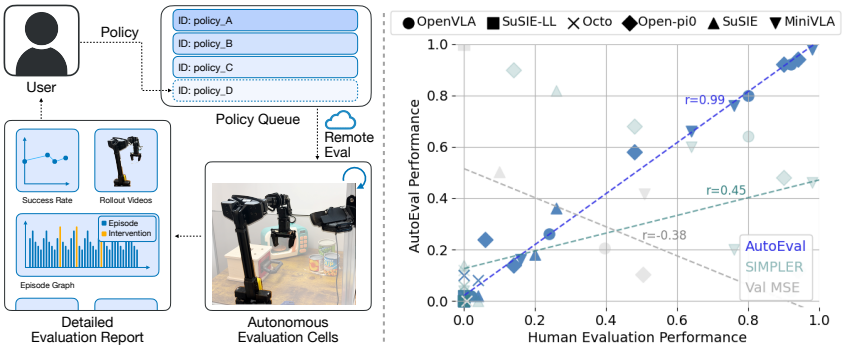


Figure 1: We introduce AutoEval, a system for scalable, automated real robot evaluation of generalist robot policies. Automated evaluation results closely match human-run evaluations, while providing a more reliable performance signal than prior simulated evaluation approaches or offline metrics. AutoEval 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.

Abstract: 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 submit evaluation jobs to AutoEval, much like how software jobs are submitted to a cluster scheduling system, and AutoEval will schedule the policies for autonomous evaluation within a framework supplying automatic success detection and scene resets. We show that AutoEval can nearly fully eliminate human involvement in the evaluation process, permitting around the clock evaluations, and 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 at https://auto-eval.github.io. In the future, we hope that AutoEval scenes can be set up across institutions to form a diverse and distributed evaluation network.

Keywords: Robot Evaluation, Generalist Policies, Robot Learning, Manipulation

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 [1, 2, 3, 4, 5, 6, 7, 8, 9], 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, evaluating robot foundation models is hugely expensive because it requires 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 [4] against its baselines required more than 2,500 rollouts across four robot setups 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 [10], 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 submit policies for evaluation as "software jobs", which gets queued and evaluated autonomously by the AutoEval system, which runs the policy, evaluates success, and resets the scene. After finishing, AutoEval returns a detailed evaluation report to the users. 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.

Designing an effective system for autonomous real-world evaluation of robot policies poses several key challenges, particularly around autonomously and reliably resetting scenes and detecting task success. Our work leverages large pre-trained models and adapts them to the evaluation scene to *learn* automatic reset policies and success detectors with high reliability.

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 [11]. 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 evaluation throughput with minimal human time, enabling 850 evaluation episodes per 24-hour period and saving $> 99\%$ human time. 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 code and a detailed step-by-step guide for setting up new AutoEval platforms within hours. We also open access to multiple Bridge-AutoEval cells at `https://auto-eval.github.io`, enabling researchers everywhere to evaluate their policies on our Bridge-AutoEval systems. We hope that this will be a step towards democratizing robotics research and enabling fair comparisons of robot policies on unified evaluation setups. While our goal is *not* to build a comprehensive benchmark for robot foundation models, we hope that by reproducing this recipe at other institutions, the robotics community will over time be able to construct a distributed comprehensive evaluation benchmark for generalist policies.

2 Related Work

Generalist robot policies. There has been significant progress in robot foundation models recently [1, 4, 3, 12, 13, 14, 15, 16, 6, 17, 8, 18], fueled by large-scale robot datasets [19, 11, 20, 5]. These models are trained to perform diverse tasks (e.g., pick-and-place, cloth folding) [11, 4, 8, 18],

adapt to various scenes with different backgrounds and distractors [21, 22], and control multiple robot embodiments (e.g., quadrupeds, manipulator arms, drones) [23, 7]. 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. This calls for an evaluation method that is much more scalable.

Robot policy evaluation in the real world. Evaluating robot policies fairly and reproducibly remains difficult due to varying hardware, task setups, and metrics [24] across institutions. To address this, multiple works have proposed robot setups with reproducible components such as 3-D printed objects or cheap robot hardware [25, 11, 26, 27, 28, 29, 30, 31, 32]. 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 with the same set of objects and hardware. Others have built evaluation systems that are hosted at a central location to compare different approaches [33, 34, 35, 24, 36, 37, 38]. 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. Our approach, AutoEval, can substantially improve the throughput of real robot evaluations by replacing parts traditionally completed by humans with specialized learned components, thus enabling robots to "evaluate themselves" 24/7. Notably, Bauer et al. [39] 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. Other existing efforts toward automating real robot evaluations have also been limited to task-specific solutions that often involve instrumenting the environment, e.g., with spring-driven or scripted reset mechanisms [40, 41, 12]. In contrast, AutoEval is designed as a general recipe for autonomous evaluation of *generalist* policies on a wider range of tasks (e.g., pick-place, articulate object & cloth manipulation) via *learned* reset and scoring modules.

Evaluation in simulation. While real-world evaluations are the gold standard, they require significant human effort and don't scale well. *Simulation* offers a scalable alternative and is widely used in robot learning [42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]. However, there are still discrepancies remain, making simulated evaluation different from real-world evaluation. First of all, physics of contacts, collisions, and friction are hard to simulate accurately [58, 59, 60, 61, 62, 63, 64, 65, 66]. Policies that interact with real-world objects usually exhibit different behavior than they do on their simulated counterparts. Secondly, real-world factors like sensor noise and delays are often minimal or ignored in simulation. Finally, visual differences (e.g., texture, lighting) make the two types of evaluation quite different [67, 68]. Recent works have tried to reduce the visual discrepancy by building realistic simulators [69, 10]. SIMPLER [10] constructs high-fidelity replicas of real robot evaluation scenes and demonstrates strong correlation between simulated and real evaluations. 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 robots and 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.

Autonomous robot operations. Multiple prior works identified the need for human supervision as a key limiting factor in robot learning [21, 70, 12, 71, 72, 73]. 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, and even some of the metrics are shared [74]. However, to our knowledge, our work is the first to explore the design of a general system for autonomous evaluation of generalist policies.

3 Autonomous Evaluation of Robot Policies in the Real World

Given a robot policy $\pi(a|o, l)$ that outputs actions under observation o and language instruction l, and a task definition $T : \mathrm{S} \to \{0, 1\}$ that maps states to task success, The policy evaluation problem is to estimate the probability of success of robot policy π in completing the task T. Typically, the policy is evaluated multiple times, while applying randomizations to the initial state of the robot and environment, to get an estimate of the policy's performance under the initial state distribution $\rho(s)$.

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. 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.

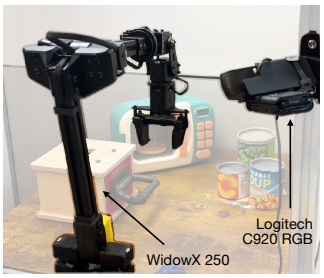
Algorithm 1 Autonomous Policy Evaluation

1: **Input:** Task T, policy π for evaluation, initial state distribution $\rho(s)$, success classifier $\mathrm{C_T}$, reset policy π_T, reset classifier $\mathrm{C}_{\rho(\mathrm{s})}$
2: **Output:** Estimated success prob. for task T
3: **for** each trial **do**
4: **Initial State:** Start with $s_0 \sim \rho(s)$
5: **Policy Rollout:** Rollout π for K steps
6: **Success Check:** Label success $\mathrm{C_T}(s_K)$
7: **Reset Scene:** Rollout reset policy π_T to return initial state to $\rho(s)$
8: **Failure:** If unable to reset or robot unhealthy, notify human operator to help
9: **end for**

Success classifier. The success classifier $\mathrm{C_T} : \mathrm{S} \to \{0, 1\}$ 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 [75, 40], AutoEval *trains* a learned success classifier $\mathrm{C_T}$, 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. 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 [76] for training the success classifier, but many other open-source VLMs would be suitable. More detailed information is provided in Appendix I.

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. To learn a reset policy, we manually collect a small set of approximately 50-100 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 (e.g., OpenVLA [4]) 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, such as robot hardware failures, damage to scene or robot, or out-of-reach objects. In AutoEval we use multiple measures to prevent and gracefully handle such issues. First, we implement a safety workspace boundary to which the robot is constrained, so poor-performing policies 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 collision. We also train a "reset suc-



(a) Bridge-AutoEval cell with a WidowX 250 6-DoF arm.

(b) Five autonomous evaluation tasks across three scenes: two pick-and-place tasks in `sink`, two articulate object manipulation tasks in `drawer`, and one deformable object manipulation task in `cloth`.

Figure 2: Bridge-AutoEval evaluation setup and task environments.

cess 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. 7).

Setup time. 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 K for easy reproduction.

4 Bridge-AutoEval: Open-Source Automated Eval Platform

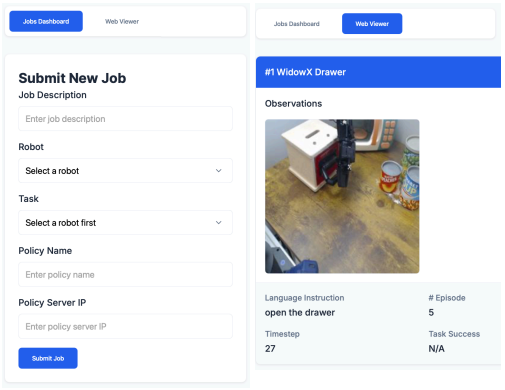
In this section, we describe an instantiation of AutoEval for multiple tasks and environments from the BridgeData V2 setup [11, 77]. BridgeData is a diverse manipulation dataset containing $60k$+ manipulation demonstrations with a WidowX 6DoF robot arm, that span 13 different skills and 24 environments. State-of-the-art generalist manipulation policies like OpenVLA [4], RT2-X [1], CrossFormer [7], and Pi0 [8, 18] are all trained on BridgeData or a super set of it [19], and therefore this setup is a natural testbed for scalable evaluation approaches for generalist policies.

We built three Bridge-AutoEval cells that works in parallel, as shown in Fig. 2b, which we call the `drawer` scene, the `sink` scene, and the `cloth` scene. We designed five tasks in total: `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 BridgeData tasks, and have been used in prior works to evaluate generalist policies [4, 21, 78, 79]. We choose these tasks since they represent diverse styles of manipulation tasks: pick-and-place, articulate object manipulation, and deformable object manipulation.

One contribution of our work is that we make two of our Bridge-AutoEval cells **publicly available** at `https://auto-eval.github.io`, so other researchers can evaluate their policies. We hope that over time, this can contribute to making evaluations in robotics more reproducible and comparable. We provide a public web UI to access our Bridge-AutoEval cells and monitor evaluation progress, as shown in Fig. 3a. Users can choose the task on which they want to perform evaluation, and provide the IP address for a "policy server" where they host their policy to be evaluated. Bridge-AutoEval will automatically queue the jobs for evaluation, and query the policy server for robot actions when the policy evaluation is executing. 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, etc. Fig. 3b shows part of an example report, which is accessible online after AutoEval finishes.

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



(a) 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.

(b) Excerpt from an AutoEval result report, provided to users upon completion of the automated evaluation. Users can see the quantitative metrics such as (per-episode) success rate, and qualitative rollout videos as well as initial and final frames to obtain a holistic understanding of the policy's performance.

Figure 3: Overview of how users interact with 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?

5.1 Experimental Setup

Tasks. We evaluate policies on the five Bridge V2 [11] 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. Metrics for success is provided in Appendix C. All tasks are performed using a WidowX 6-DoF robot arm.

Policies. We run evaluations with six recently released generalist robot policies: **OpenVLA** [4], **Octo** [80], **Open-π_0** [81, 8], **MiniVLA** [82], **SuSIE** [79], and **SuSIE-LL** [79]. See more detailed descriptions of these policies in Appendix C. This set of policies is a representative sample of current state-of-the-art generalist policies. All policies contain Bridge V2 as part of their training data.

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** [10]. SIMPLER builds realistic simulated versions of real-world environments and evaluates policies purely in simulation. We reuse the existing SIMPLER environment for the Bridge sink environment, and build a new SIMPLER simulation environment for the drawer scene (Fig. 4) by following Li et al. [10]'s step-by-step guide. Deformable objects such as cloth are hard to simulate in general [83, 84], and at the time of writing the simulator of SIMPLER, Maniskill [85], 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.

Figure 4: SIMPLER [10] 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.

Metrics. For scalable evaluation approaches, the goal is to approximate the result of human-run evaluations, the gold standard for robotic policy evaluation, as closely as possible. Following Li et al. [10] we use two metrics to measure how closely respective evaluation results match those of

human-run evaluations: (1) **Pearson correlation** [86], a widely used statistical tool that measures the linear consistency between two random variables, with scores nearing 1 indicating high correlation; (2) **MMRV** (Mean Maximum Rank Violation) [10], which measures the consistency of policy *ranking* and, as described in Li et al. [10], can be more robust to noise on the evaluation results. Low MMRV scores (See Appendix D for details) indicates closely matching evaluation results.

5.2 AutoEval Closely Matches Human Evaluation Results

To investigate how well the different evaluation appraoches from Section 5.1 match human-run results, we run 50 evaluation rollouts for each policy in each task (except "val-MSE", which does not require rollouts). We report results in Fig. 5, with a detailed breakdown of results per task, policy, and evaluation method in Appendix, Table 1 to Table 3. Similar to prior work [10], 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.

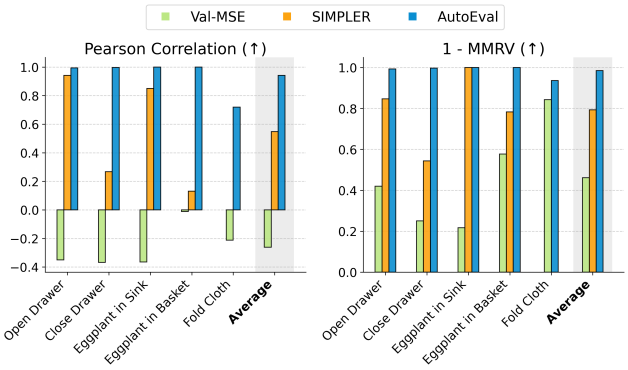


Figure 5: **Correlation of scalable evaluation approaches to human-run evaluations.** AutoEval closely matches human evaluations, achieving high correlation and low MMRV (plotted in the figure is $1 - \text{MMRV}$ for clarity). In contrast, SIMPLER simulated evaluations and validation MSE do not correlate as well with human evaluations.

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. In particular, an MMRV score close to zero indicates that it rarely disrupts the ranking of policies. Intuitively, since evaluations are 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; while there are occasional failures, the accuracy of AutoEval is sufficient to provide a strong ranking signal. We show qualitative examples of autonomous evaluation rollouts in Fig. 6, and further examples in Appendix B.

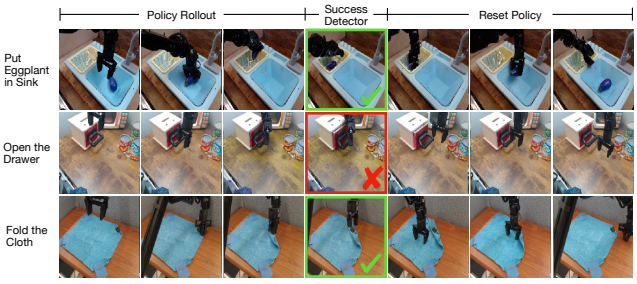


Figure 6: Qualitative visualization of AutoEval evaluation rollouts on three of our tasks. After policy rollout, the success classifier determines whether the rollout was successful. Then, the reset policy returns the environment to the initial state distribution for the next evaluation. Our evaluations cover representative robot manipulation tasks: pick-place, articulate, and deformable object manipulation.

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.

AutoEval saves 99%+ human time. We performed a long-running evaluation over 24 hours, interleaving the evaluation of various policies, using the "*open drawer*" and "*close drawer*" tasks. In Fig. 7, we present the evaluation throughput, as well as the number of human interventions needed over the span of the whole 24 hours. We report evaluation throughput as valid evaluation steps per minute, excluding reset policy steps. In Fig. 7, the throughput varies based on the inference speed of different policies. On average, the single AutoEval cell is able to run 42 evaluation steps per minutes, totaling roughly 850 episodes on `drawer`. The average AutoEval speed, shown in dotted blue line, is slightly lower but on par with the average evaluation speed of a human 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. For each intervention, the human operator simply needed to reset the objects in the scene, and potentially reset the robot arm in case of a non-recoverable motor failure. Assuming that each human reset operation takes 1 minute, AutoEval reduces human time involvement by $> 99\%$ (Detailed calculation in Appendix H).

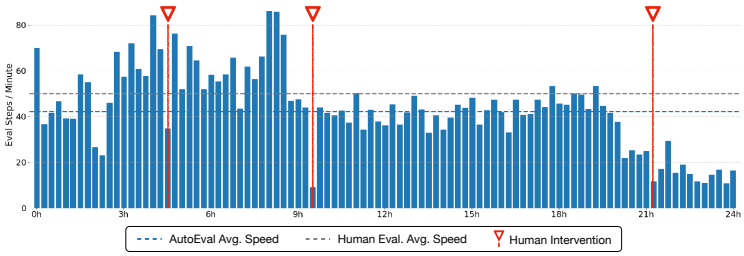


Figure 7: Visualization of a 24 hour AutoEval evaluation run with ˜850 total evaluation episodes. AutoEval is able to run autonomously and only required a total of 3 human interventions over a 24 hour period. On average, the evaluation throughput of AutoEval is on par with that of human evaluations, but saves **99%+** human operator time.

AutoEval results consistent across time. We test AutoEval's ability to produce comparable performance estimates across multiple evaluations of the same policy. Specifically, we evaluate Open-π_0 for 9 times in a row on the "open drawer" task, each with 50 individual trials, totaling 450 trials over 11 hours. Results in Fig. 8 show that AutoEval produces consistent evaluation results across long periods of time. Concretely, for the first 7 evaluations, AutoEval results are consistent within the natural variance of robot evaluations ($\pm 10\%$). We notice a regression in performance after approximately 8 hours of continuous operation, which we attribute to motors overheating on our affordable WidowX robot ($<$ \$3500) after long hours of operation. To mitigate such overheating in practice, we pause autonomous evaluations for 20 minutes every 6 hours to let the motors cool off.

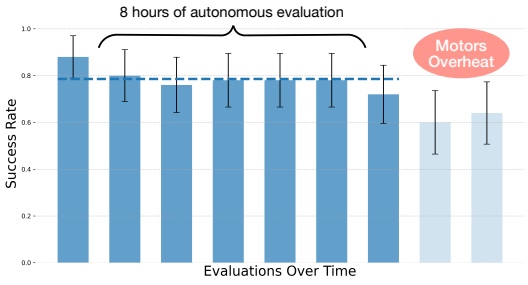


Figure 8: AutoEval results remain consistent over 8 hours of autonomous evaluations. Error bars show 95% confidence intervals. After 8 hours, motors overheat and affects evaluation performance; we therefore let motors rest every 6 hours.

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

6 Conclusion

We introduced AutoEval, a system for autonomous evaluation of generalist robot policies in the real world. AutoEval can perform evaluations around the clock and with minimal human involvements across a range of commonly used robot evaluation tasks, and evaluation results closely matches those of human-run evaluations. It is both more reliable and applicable to a wider range of tasks than prior simulation-based evaluation approaches. To make real-robot evaluation widely available and more comparable, we provide public access to two AutoEval evaluation cells for BridgeData V2 evaluation tasks. Users can submit 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.

Limitations

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 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. 9. 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.

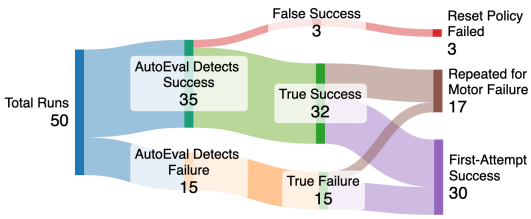


Figure 9: Analyzing 50 AutoEval runs on the `sink` scene: the main failure mode is false positive results because the reset policy failed.

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.

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. [87] 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 table-top 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.

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 10 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 Robot Setup and Experiment Details

Similarly to Walke et al. [11], our Bridge-AutoEval 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. 2a. We use end effector delta action with blocking control. We maintain constant lighting over each robot station.

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 consider the following generalist policies for evaluation in Section 5.1: **OpenVLA** [4], a 7B parameter vision-language-action model (VLA) pre-trained on the Open X-Embodiment dataset [19], **Octo** [80], a 27M parameter transformer policy, also pre-trained on Open X-Embodiment, **Open-π_0** [81], an open-source reproduction of the 3B parameter π_0 VLA [8] (the original π_0 was not available in open-source at the time of writing), pre-trained on the Bridge V2 dataset, **MiniVLA** [82], a 3B parameter VLA pre-trained on the Bridge V2 dataset [11], **SuSIE** [79], 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. We evaluate the publicly released checkpoints for all models.

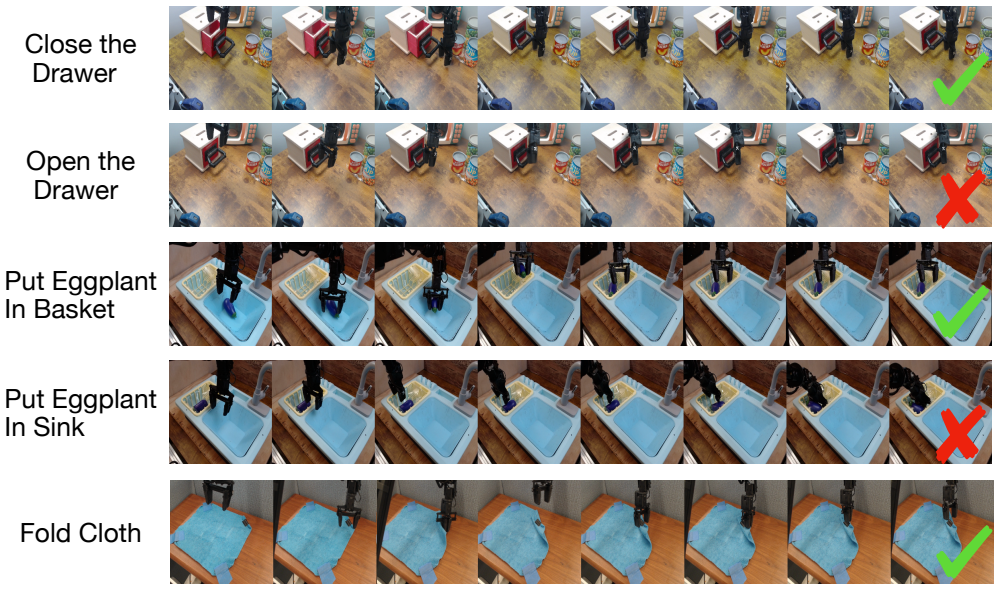


Figure 10: Samples of autonomous policy evaluation trials with AutoEval on the five tasks. The classifier result of each task is visualized on the right hand side, and the reset policy is not shown.

D MMRV Score Computation

Following Li et al. [10], 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:

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

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

E 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.

F Evaluation on Bridge-SIMPLER [10]

In AutoEval , we introduced a new Drawer Scene to the existing SIMPLER [10] setup for the WidowX robot. The scene was visually matched with the AutoEval's Drawer setup, and overlaid

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.

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 [10] 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 [10] for Drawer and Sink Scene on four tasks and six different policies.

G Computing Action Validation MSE between policies

We sample 400 trajectories from the validation set of BridgeData [11] to compute the action mean squared error (MSE) for each policy. The results are shown in Table 4. Consistent with the findings in SIMPLER [10], 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.

H Details on Reducing Human Involvement

Here we detail how we arrived at the $99\%+$ human time reduction. 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 ≈ 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 ≈ 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 $> 99\%$.

I 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.

J 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 $> 95\%$.

K 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_evalfor 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 [4], 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 [76] 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.

L 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. 8 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 [82] on the same demos. We find that all reset policies have success rate $> 95\%$.

M 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.

N 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. 11 (50 trials each). Qualitatively, we find that the reset distributions of other tasks are similarly overlapping, and also roughly cover the task distribution.

Figure 11: Initial state distribution for 3 different AutoEval runs is consistent. Red dots show the centroid position of the eggplant. Each run uses 50 trials.

O 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. 9). 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.

References

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

[2] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

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

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

[5] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. 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.

[6] A. Sridhar, D. Shah, C. Glossop, and S. 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.

[7] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

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

[9] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[10] 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.

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

[12] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[13] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, et al. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv preprint arXiv:2312.02976*, 2023.

[14] 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.

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

[16] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[17] S. Ye, J. Jang, B. Jeon, S. Joo, J. Yang, B. Peng, A. Mandlekar, R. Tan, Y.-W. Chao, B. Y. Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

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

[19] O. X.-E. Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, H. Su, H.-S. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Kim, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Majd, K. Rana, K. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. hua Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

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

[21] Z. Zhou, P. Atreya, A. Lee, H. Walke, O. Mees, and S. Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

[22] L. Fu, H. Huang, G. Datta, L. Y. Chen, W. C.-H. Panitch, F. Liu, H. Li, and K. Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

[23] J. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, D. Sadigh, and S. Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.

[24] K. Van Wyk, J. Falco, and E. 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.

[25] B. Yang, J. Zhang, D. Jayaraman, and S. Levine. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *ICRA*, 2019.

[26] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *arXiv preprint arXiv:2305.12821*, 2023.

[27] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, page 02783649241276017, 2023.

[28] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

[29] S. Tyree, J. Tremblay, T. To, J. Cheng, T. Mosier, J. Smith, and S. 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.

[30] J. Leitner, A. W. Tow, N. Sünderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. 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.

[31] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. 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.

[32] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. 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.

[33] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski. The darpa robotics challenge finals: Results and perspectives. *The DARPA robotics challenge finals: Humanoid robots to the rescue*, pages 1–26, 2018.

[34] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[35] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347, 1997.

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

[37] 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)*, pages 9197–9203. IEEE, 2023.

[38] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. W. Clegg, J. Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

[39] 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*, pages 190–204. PMLR, 2022.

[40] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[41] D. B. D'Ambrosio, S. Abeyruwan, L. Graesser, A. Iscen, H. B. Amor, A. Bewley, B. J. Reed, K. Reymann, L. Takayama, Y. Tassa, et al. Achieving human level competitive robot table tennis. *arXiv preprint arXiv:2408.03906*, 2024.

[42] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[43] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[44] Y. Lee, E. S. Hu, and J. 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.

[45] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[46] 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.

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

[48] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[49] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-k. Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

[50] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[51] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023.

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

[53] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, and S. Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

[54] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, W. Ai, B. 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.

[55] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[56] 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.

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

[58] E. Todorov, T. Erez, and Y. 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.

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

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

[61] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. Karen Liu. Dart: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22): 500, 2018.

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

[63] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. 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.

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

[65] P. Huang, X. Zhang, Z. Cao, S. Liu, M. Xu, W. Ding, J. Francis, B. Chen, and D. Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In *Conference on Robot Learning*, pages 734–760. PMLR, 2023.

[66] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[67] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. 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.

[68] 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. *IEEE Robotics and Automation Letters*, 4(2):1148–1155, 2019.

[69] Z. Li, T.-W. Yu, S. Sang, S. Wang, M. Song, Y. Liu, Y.-Y. Yeh, R. Zhu, N. Gundavarapu, J. 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.

[70] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, S. Maddineni, K. Rao, D. Sadigh, P. Sanketi, P. Sermanet, Q. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

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

[72] A. S. Chen, H. Nam, S. Nair, and C. Finn. Batch exploration with examples for scalable robotic reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4401–4408, 2021.

[73] T. Lampe, A. Abdolmaleki, S. Bechtle, S. H. Huang, J. T. Springenberg, M. Bloesch, O. Groth, R. Hafner, T. Hertweck, M. Neunert, et al. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. *arXiv preprint arXiv:2312.11374*, 2023.

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

[75] S. Gu, E. Holly, T. Lillicrap, and S. 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.

[76] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[77] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[78] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[79] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[80] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

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

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

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

[84] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[85] 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. *arXiv preprint arXiv:2107.14483*, 2021.

[86] K. 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.

[87] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh. A taxonomy for evaluating generalist robot policies. *arXiv preprint arXiv:2503.01238*, 2025.