RoboArena: Distributed Real-World Evaluation of Generalist Robot Policies

Pranav Atreya[*,1] **Karl Pertsch**[*,1,2] **Tony Lee**[*,2]
Moo Jin Kim[2] **Arhan Jain**[3] **Artur Kuramshin**[4] **Clemens Eppner**[5] **Cyrus Neary**[4] **Edward Hu**[6]
Fabio Ramos[5] **Jonathan Tremblay**[5] **Kanav Arora**[3] **Kirsty Ellis**[4] **Luca Macesanu**[7] **Matthew Leonard**[6]
Meedeum Cho[8] **Ozgur Aslan**[4] **Shivin Dass**[7] **Jie Wang**[6] **Xingfang Yuan**[6] **Xuning Yang**[5] **Abhishek Gupta**[3]
Dinesh Jayaraman[6] **Glen Berseth**[4] **Kostas Daniilidis**[6] **Roberto Martin-Martin**[7] **Youngwoon Lee**[8]
Percy Liang[2] **Chelsea Finn**[2] **Sergey Levine**[1]

https://robo-arena.github.io

Abstract:
Comprehensive, unbiased, and comparable evaluation of modern generalist policies is uniquely challenging: existing approaches for robot benchmarking typically rely on heavy standardization, either by specifying fixed evaluation tasks and environments, or by hosting centralized "robot challenges", and do not readily scale to evaluating generalist policies across a broad range of tasks and environments. In this work, we propose RoboArena, a new approach for scalable evaluation of generalist robot policies in the real world. Instead of standardizing evaluations around fixed tasks, environments, or locations, we propose to crowd-source evaluations across a distributed network of evaluators. Importantly, evaluators can freely choose the tasks and environments they evaluate on, enabling easy scaling of diversity, but they are required to perform double-blind evaluations over *pairs* of policies. Then, by aggregating preference feedback from pairwise comparisons across diverse tasks and environments, we can derive a ranking of policies. We instantiate our approach across a network of evaluators at seven academic institutions using the DROID robot platform. Through more than 600 pairwise real-robot evaluation episodes across seven generalist policies, we demonstrate that our crowd-sourced approach can more accurately rank the performance of existing generalist policies than conventional, centralized evaluation approaches, while being more scalable, resilient, and trustworthy. We open our evaluation network to the community and hope that it can enable more accessible comparisons of generalist robot policies.

Keywords: Generalist Robot Policy Evaluation, Crowd-Sourced Evaluation

1 Introduction

Modern robot policies are increasingly general, and able to perform a wide range of tasks across many environments [2, 3, 4, 5, 6]. With the increased generality a new challenge arises: how can we accurately measure and compare the performance of such generalist policies? Conventional approaches to robot evaluation and benchmarking rely on comparing policies across tightly standardized sets of environments and tasks [7, 8, 9, 10], and are typically restricted to only a handful of tasks across a small number of scenes [11, 12, 13, 14]. As such, they are not well-suited to

[*]: Core contributors, detailed contributions in Appendix A

Correspondence to: pranavatreya@berkeley.edu, pertsch@berkeley.edu, tonyhlee@stanford.edu
[1]UC Berkeley, [2]Stanford University, [3]University of Washington, [4]University of Montreal, [5]NVIDIA, [6]University of Pennsylvania, [7]UT Austin, [8]Yonsei University

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

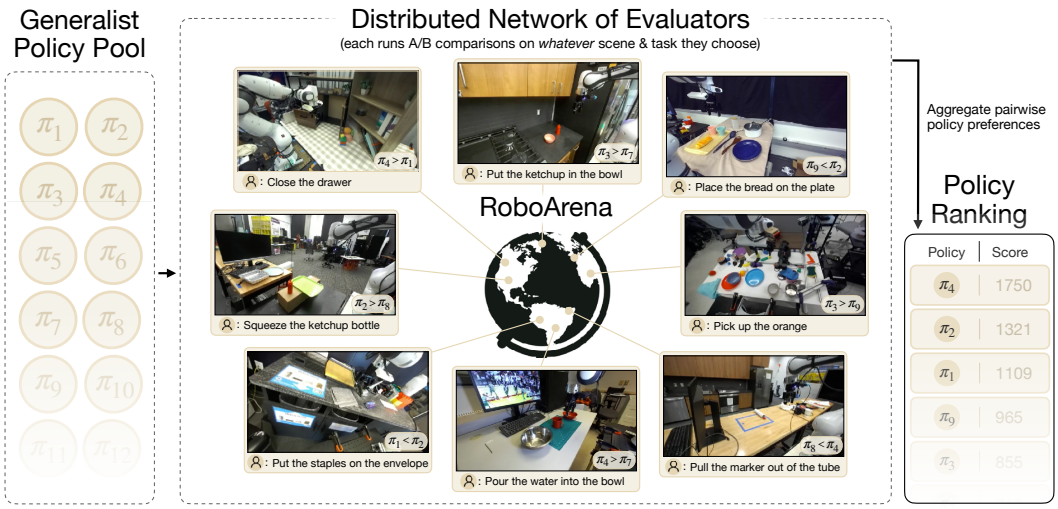

Figure 1: We present RoboArena, a distributed real-world evaluation framework for generalist robot policies. Instead of standardizing environments and tasks, RoboArena aggregates crowd-sourced pairwise A/B policy evaluations across a broad spectrum of environments and tasks to derive a global policy ranking. Its decentralized design makes RoboArena a scalable, comprehensive, and trustworthy framework for generalist robot policy evaluation. We open-source an instantiation of RoboArena on the DROID robot platform [1] and invite community members to participate, both by contributing policies and running evaluations.

provide comprehensive performance evaluations for policies that are designed to perform across a much broader spectrum of initial conditions. At the same time, existing evaluation approaches are challenging to scale: guaranteeing comparable conditions across a large number of real-world tasks and environments faces many practical challenges, from accurately reproducing scene layouts and lighting conditions, to maintaining large, centralized fleets of evaluation robots, and manufacturing differences between said robots. As such, the development of increasingly general robot policies requires us to rethink the way we evaluate robot policies.

At first glance, broadly capable robot policies seem to compound the reproducibility and scalability challenges faced by robot evaluations and benchmarks in the past. Yet, in this work, we argue that increasingly general robot policies offer an *opportunity* for an alternative approach to robot evaluation, that has the promise to address many of the challenges of prior evaluation efforts. Realizing this potential, however, requires rethinking how we run robot evaluations: instead of *standardization* and centralized evaluation "challenges", we argue that *decentralization* and an evaluation approach that embraces the non-stationarity of the physical world offer a promising alternative.

To this end, we propose RoboArena, a distributed real-world evaluation framework for generalist robot policies. Inspired by recent crowdsourced benchmarks for evaluating generalist language or vision-language models like Chatbot Arena [15] or GenAI Arena [16], RoboArena relies on a decentralized network of evaluators that perform *pairwise*, *double-blind* comparisons of policies in *whichever scene* and on *whatever task* they deem suitable. The evaluator then provides a preference for which of the two policies performed better, along with a free-form language explanation. Our evaluation algorithm aggregates a large number of such pairwise comparisons into a global policy ranking, as well as a set of qualitative characteristics, strengths and weaknesses for each policy. This decentralized design allows RoboArena to be **open-ended** in the number of environments and tasks policies are evaluated on, **robust**, since many entities across the decentralized network contribute via double-blind evaluations to the final score, and **scalable**, with dozens of evaluators asynchronously contributing policy evaluations at any time of day. By not relying on identical initial conditions beyond the horizon of a single pairwise policy comparison, RoboArena foregoes many of the practical challenges prior robot evaluation frameworks faced.

In this paper, we outline our distributed evaluation protocol, including algorithms for aggregating pairwise policy comparisons into global policy rankings, and for extracting qualitative policy characteristics via LLM-assisted analysis of evaluation results. We then instantiate RoboArena on the DROID robot platform [1], on which modern policies can out-of-the-box generalize to new scenes and tasks [5]. Through decentralized evaluations of 7 generalist policies across 7 universities and a total of 612 pairwise comparisons, we demonstrate that RoboArena provides more accurate performance rankings of generalist policies than the conventional, centralized evaluation scheme used in prior work, when compared to an "oracle" policy ranking computed via exhaustive evaluation of all policies on all tested tasks. At the same time, RoboArena matches the episode efficiency of standard evaluation, i.e., the same number of evaluations, when distributed across the RoboArena evaluator network, lead to higher quality policy rankings. In addition to the evaluation framework, our work also provides the most comprehensive evaluations of generalist robot policies to date (4284 evaluation episodes across numerous tasks and scenes), and highlights the limitations of current policies.

2 Related Work

Simulated and real-world robot evaluation. Simulated evaluations offers perfect reproducibility, and can be efficiently parallelized. Thus, numerous simulated robot benchmarks have been proposed over the years [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]. However, simulated environments are an imperfect replica of the real world, and even works that optimize for visual and physical fidelity [29] are often limited in the diversity of tasks they support, and sometimes fail to accurately reflect real-world policy performance [30]. In the real world, benchmarks try to ensure reproducibility through detailed manuals for reproducing environment setups [7, 8, 9, 10, 31, 32], enabling remote access to centrally hosted evaluations [33, 12, 13, 30], or organizing in-person challenges [34, 11, 35, 14, 36]. However, to ensure reproducibility of initial conditions and make centralized evaluation feasible, these approaches typically restrict evaluations to a narrow set of tasks and environments, or even a specific challenge date. In this work, we propose an alternative approach for real-world robot evaluation based on crowd-sourcing pairwise policy comparisons across a *decentralized* network of evaluators. Our approach can cover a wider distribution of tasks and environments, and is inherently more resilient and scalable than existing, centralized evaluation approaches.

Generalist robot policies. Recently, robotics has seen a trend towards training *generalist* robot policies [37, 38, 2, 39, 40, 3, 41, 42, 43, 44, 45, 5, 6, 46, 47] on diverse datasets of robot experience [48, 49, 50, 2, 1, 51, 52, 53, 54]. Notably, multiple works have demonstrated policies that can perform tasks across many environments *out of the box* [55, 56, 4, 5]. At the same time, conventional approaches to robot evaluation struggle to comprehensively evaluate the performance of such generalist policies, since they are cumbersome to scale to broad distributions of tasks and environments, motivating our distributed evaluation approach in this work.

Crowd-sourced benchmarks. In recent years, crowd-sourced benchmarks have gained popularity, from language modeling [15], to generative image and video modeling [16], or even for specialized legal model training [57]. In robotics, Dasari et al. [58] proposed a benchmark that aggregates policy ranking results across institutions to obtain more comprehensive evaluations of learning *algorithms*, but it required retraining of policies for every new institution and task at hand, and thus remained limited to a small number of tested environments and tasks. In contrast, our work proposes a distributed benchmark for *generalist robot policies*, which can be evaluated out-of-the-box, and thus makes it feasible to evaluate on a much broader distribution of tasks and environments.

3 Decentralized Robot Policy Evaluation via Pairwise Comparison

Conventional robot policy evaluations aim to *standardize* the conditions under which policies are compared as much as possible, typically by defining fixed sets of tasks and scenes policies should be run on, and by trying to closely match initial conditions like lighting, camera angles, or scene

background [7, 9, 50, 59]. As we discussed in Section 1, reproducing such standardized conditions in the real world is extremely challenging, particularly when evaluations should be run not just in one central place, but across institutions.

In this section, we introduce an alternative approach for robot evaluation that foregoes repeatedly evaluating policies in standardized settings, and instead relies on *decentralized, pair-wise, double-blind policy comparisons*. Crucially, our evaluation approach *does not* prescribe what scene or task a policy pair should be evaluated in, but instead aggregates a large number of pairwise comparisons across *many* tasks and scenes to establish a policy ranking. This decentralized approach has multiple benefits: it is **open-ended**, since we do not standardize tasks and environments, thus broadening coverage; it is **robust**, since no single entity can easily sway results in double-blind, decentralized evaluations; it is **scalable**, since many institutions can collaborate on the evaluations; and it is **adaptable**, since tasks can naturally adapt to the frontier of policy capabilities.

In the remainder of this section, we will first describe our evaluation procedure (Section 3.1), then detail how we aggregate pairwise comparisons into global policy rankings (Section 3.2), and finally describe tools for extracting qualitative policy characteristics from the evaluation data (Section 3.3).

3.1 Policy Evaluation Protocol

We provide a summary of our policy evaluation protocol in Algorithm 1. Given a pool of policies $\pi_{1\ldots N} \in \Pi$, our goal is to estimate a performance ranking. We design our evaluation for *generalist* robot policies, and thus assume that all policies in Π can be meaningfully (and safely) evaluated across a broad range of scenes and tasks. Additionally, we assume access to a pool of evaluators E that asynchronously run real robot evaluations, and a central evaluation server C that manages the decentralized evaluation operation.

Algorithm 1 Policy Evaluation Protocol

Require: Set of policies Π, Evaluator E, Central Server C
Ensure: Global policy ranking, policy characteristics
1: **for** $i = 1$ to K evaluations **do**
2: E requests policies for evaluation from C
3: C samples two policies $\pi_A, \pi_B \sim \Pi$
4: E rearranges scene and defines task T_i
5: E executes π_A and π_B sequentially
6: E provides pairwise feedback $F_i(\pi_A, \pi_B)$ to C
7: C aggregates $\{F_i\}_{i=1}^{K}$ into global ranking ▷ Section 3.2
8: C extracts qualitative policy characteristics ▷ Sec. 3.3

During an evaluation session, an evaluator E requests two policies from the central server C. C randomly samples two policies $(\pi_A, \pi_B) \in \Pi$ and assigns them to E. To ensure unbiased evaluation, the evaluators do not know which policies they are evaluating. In practice, we simply provide them with the IP addresses of remotely hosted evaluation servers. After policies are sampled, the evaluator arranges the evaluation scene, e.g., by moving the robot to a new location and rearranging the objects in front of the robot, and defines the evaluation task T_i in form of a natural language instruction. Then, E runs rollouts for policies π_A and π_B back-to-back until the task is completed or a fixed timeout is reached. Importantly, we require the evaluator E to *closely match* the initial conditions *within* this A/B policy comparison (while they can choose to change them *between* separate pairwise evaluations). This ensures that the comparison of policies π_A and π_B is fair.

After both evaluations are complete, E provides feedback $F(\pi_A, \pi_B)$ about the performance of the policies. **We ask evaluators to provide three types of feedback:** a continuous **progress score** $\in [0 \ldots 100]$ that is proportional to the maximum *progress* a policy achieved on the task (e.g., 0 for no progress, 100 for successfully executing the task, intermediate values for partial success); a binary, **pairwise preference** label that indicates which policy the evaluator preferred (we leave it to the evaluator to decide how to determine their policy preference); and a free-form, **natural language explanation** for *why* they preferred one policy over the other.

After the task instruction T, pairwise feedback F, and recordings of all observations and actions are sent to the central server C, the evaluator may choose to continue with another evaluation session, or pause and return at a later time. All evaluations outside a single pairwise comparison can run fully *asynchronously* at any time or place.

3.2 Computing Global Policy Rankings

In this section, we discuss our algorithm for computing a global policy ranking using the pairwise feedback provided by the evaluators. Formally, we are given a set of N policies $\Pi = \{\pi_1, \ldots, \pi_N\}$ and a dataset of pairwise preferences $\mathcal{D}_p = \{P_{\pi_A,\pi_B}, t\}$, where $P_{\pi_A,\pi_B} \in \{0, 1\}$ indicates a binary preference, and t identifies the task the A/B evaluation was run on (e.g., a specific scene and language instruction). We aim to compute a global policy ranking $\mathcal{R} : \pi_i > \pi_j > \cdots > \pi_k$.

Bradley-Terry Model The Bradley-Terry (BT) model [60] is commonly used in a learning from preferences setting. BT models the win-probability of policy π_A versus π_B as the sigmoid of the difference of log-abilities: $p(\pi_A > \pi_B) = \sigma(\theta_A - \theta_B)$. Then, the log-ability parameters of this model can be fit either via an online algorithm such as Elo [61], or in an offline setting with an iterative algorithm guaranteed to converge to the maximum likelihood fit of the data [62]. This iterative algorithm can also be modified to support ties [63].

Extending BT The Bradley-Terry model, while appealing due to the existence of stable offline solvers, assumes that each pairwise comparison in the preference dataset takes place under identical conditions. This assumption is not satisfied in the setting of A/B robot policy evaluations, where the *task* under which the A/B evaluation is conducted can vary from one evaluation to the next. Importantly, the task can significantly affect the policy preference: for example, a task that is very difficult or very easy for both policies can diminish any differentiating signal between π_A and π_B, and thus the probability that π_A is preferred over π_B by evaluators. Further, pairs of policies can exhibit different relative performance relationships on different subsets of tasks; a specialized policy for one subset of tasks would, for example, be preferred over more generalist policies on this subset, but not in aggregate. Thus, it is important to account for task-effects when modeling the preference data. Standard BT models would simply treat task-related effects as noise, leading to worse rankings.

We propose to augment the BT log-ability parameters for each of the policies $\theta = (\theta_1 \ldots \theta_N)$ with task difficulty parameters $\tau = (\tau_1 \ldots \tau_T)$, marginal task probabilities $\nu = (\nu_1 \ldots \nu_T)$ s.t. $\sum_t \nu_t = 1$, defining the prior probability any given A/B trial belongs to latent bucket t, and policy-task offsets $\psi = ((\psi_{1_1} \ldots \psi_{1_T}), \ldots, (\psi_{N_1} \ldots \psi_{N_T}))$, which model *policy-dependent* task difficulty:

$$p(\pi_A > \pi_B) = \sum_{t=1}^{T} \nu_t \cdot \sigma(\theta_A + \psi_{A_t} - \tau_t) \cdot (1 - \sigma(\theta_B + \psi_{B_t} - \tau_t)) \tag{1}$$

Critically, all parameters can be learned solely from preference data; no auxiliary task information is required. Via the maximum likelihood estimation process, the task-related parameters τ, ν, and ψ will be automatically fit. The number of task buckets T is a hyperparameter.

An EM algorithm for approximate MLE The parameters θ, τ, ν and ψ can be fit via an approximate maximum likelihood expectation-maximization (EM) algorithm. In essence, the algorithm iterates between measuring the likelihood of the data under the current model parameters, calculating the first and second-order derivatives of this likelihood, performing a maximization step with clipped Newton updates, and centering the new parameters to maintain zero mean. A detailed derivation of the derivatives and description of the EM update algorithm can be found in Appendix B.

3.3 Extracting Qualitative Policy Characteristics

Estimating *qualitative* policy characteristics, e.g., a model's ability to follow language or perform multi-step tasks, is crucial to guide future research. Typically, researchers develop an intuition for such qualitative characteristics by running all evaluations for a given policy themselves. Yet, in a distributed evaluation setup we need new tools to synthesize such nuanced insights from hundreds of policy rollout videos, task instructions, and evaluator language feedback. To this end, we experiment with using large language and vision-language models (LLMs, VLMs) to assist with the analysis.

We provide an overview of our prototype analysis tool in Figure 2. We pass the first images of each evaluation video and the corresponding task instruction to a VLM (`OpenAI GPT-4.5`) and ask it to categorize the task (e.g., pick-place vs. open-close vs. tool use) and describe the scene's lighting, clutter, object visibility, etc. Then, for each policy in our pool, we use an LLM (`OpenAI`

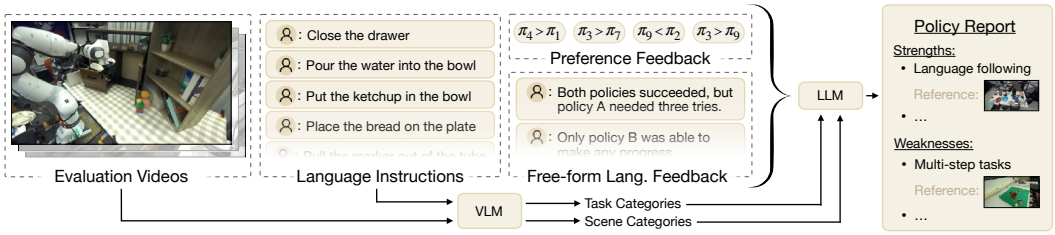

Figure 2: Pipeline for extracting *qualitative* policy characteristics from RoboArena's rich evaluation data. We use a VLMs to categorize scenes and tasks, and then use an LLM to aggregate information across a large number of evaluation rollouts into a *policy report* that summarizes qualitative strengths and weaknesses, and cites concrete evaluation rollout videos as evidence.

o3) to generate a *policy report* by summarizing preference annotations, categorization results, and free-form evaluator feedback for all evaluations. We instruct the LLM to compare performance to other policies along the task categories, and to extract qualitative policy characteristics from the language feedback. Importantly, the LLM is instructed to *cite* evaluation episodes as evidence for any claim in the report and we automatically annotate the report with videos from these rollouts to enable researchers to verify any claims. We experimentally verify our tool in Section 4.4, and include example policy reports in the supplementary data.

4 Experiments

The goal of our experimental evaluation is to answer the following questions: (1) How does RoboArena compare to conventional robot evaluation approaches for ranking the performance of generalist robot policies? (2) How *sample efficient* is RoboArena in ranking robot policies? (3) Can RoboArena extract *qualitative* insights about policy performance beyond aggregate performance?

4.1 Experimental Setup

Evaluation system. We instantiate RoboArena on the DROID robot platform [1]. DROID is well-suited for prototyping RoboArena, since it is open-source, comes with a large, diverse dataset, and there are published, generalist DROID policies that we can use for evaluation. For details about our evaluation system design and open-source considerations, see Appendix C. We assemble a group of evaluators across *seven* institutions in multiple countries and continents to test RoboArena.

Policy pool. We populate the policy pool with all publicly available *generalist* DROID policies that have been shown to work out of the box in new environments. At the time of writing and to our knowledge, there are *seven* such policies, based off PaliGemma [64] or π_0 [41] base models. Concretely, we evaluate the following policies: **π_0-flow-DROID**, the π_0 flow-vision-language-action model (VLA) [41], fine-tuned on the DROID dataset; **π_0-FAST-DROID**, the π_0-FAST autoregressive VLA [5], fine-tuned on the DROID dataset; **PG-flow-DROID, PG-FAST-DROID, PG-FAST+-DROID, PG-FSQ-DROID, PG-Bin-DROID**, PaliGemma [64] vision-language models (VLMs), fine-tuned on the DROID dataset using different action representations from Pertsch et al. [5]: π_0-style flow matching, FAST/FAST+ tokenization [5], finite scalar quantization [65], and simple binning tokenization [38, 3]. We use the publicly available implementation in [66] for all policies.

Comparison. We establish an "oracle" policy ranking by exhaustively evaluating *all* policies on *all* tested tasks and comparing their average progress scores. This was done by asking evaluators to score the remaining five policies after each A/B evaluation on the same task. In total, we use 4284 evaluations to compute this oracle ranking. We compare RoboArena to a conventional robot policy evaluation approach, which tests all policies on a fixed, tightly standardized set of evaluation tasks in a narrower set of environments. **Concretely, we compare to the DROID evaluation procedure used in Pertsch et al. [5]**, which consists of 17 tasks and 44 episodes per policy and is representative of typical robot evaluations ([3, 67], see the appendix of Pertsch et al. [5] for a detailed list of tasks).

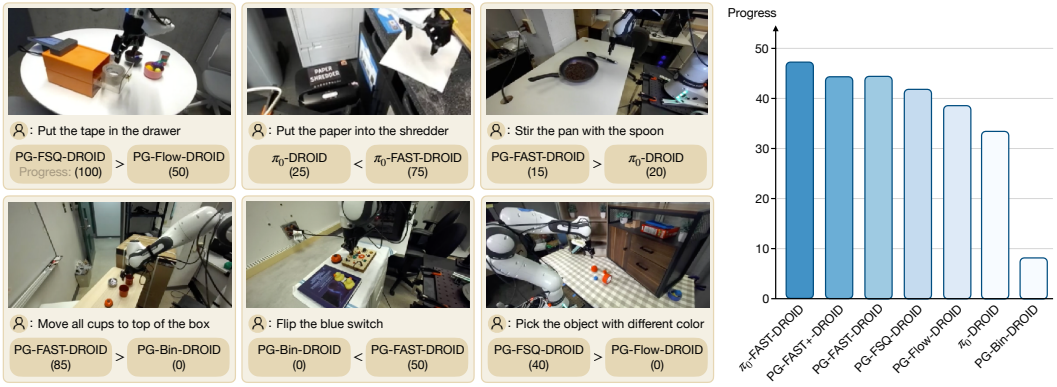

Figure 3: **Left**: Examples of RoboArena evaluations. Evaluations span a diverse set of scenes and tasks. **Right**: "Oracle" policy ranking, aggregated from progress scores of 4284 evaluation rollouts.

4.2 RoboArena Accurately Ranks Policy Performances

In total, our RoboArena evaluation covers 4284 policy rollouts across dozens of scenes and hundreds of task instructions. We show a few examples in Figure 3, left, and many more in Appendix D. To our knowledge, this is the most extensive evaluation of generalist policies to date. While the remainder of this section is devoted to evaluating our RoboArena evaluation framework itself, we provide a detailed analysis of the underlying evaluation data, including strengths and weaknesses of current generalist policies, in Appendix G.

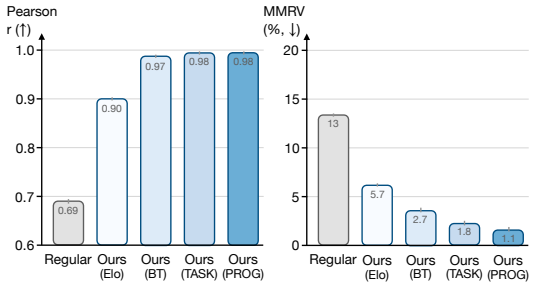

Figure 4: Policy rankings from RoboArena pairwise comparisons correlate significantly better with oracle rankings than conventional robot evaluation approaches ("Regular"). Our task-aware ranking approach ("TASK") leads to the best ranking. Ranking with progress scores ("PROG") also proves effective, but may lose more nuanced information about policy performance.

We evaluate how well our pairwise evaluation approach can approximate the oracle ranking (see Figure 3, right), which is computed through exhaustive evaluation of all policies on all tasks. We compare RoboArena to conventional robot evaluations from prior work. We follow Li et al. [29] and report Pearson correlation r, as well as Mean Maximum Rank Violation (MMRV), a ranking metric that takes the *performance difference* between policies into account.

We test instantiations of RoboArena with standard ranking algorithms, namely Elo [61] and Bradley-Terry ("BT", [60]), and our task-aware ranking approach introduced in Section 3.2 ("TASK"). The results in Figure 4 show that conventional robot evaluations ("Regular"), which due to reproducibility challenges are restricted to a relatively small number of tasks and environments, do not provide a reliable performance estimate for *generalist* policies. **This highlights the benefit of evaluations *without* task and environment standardization, as they lead to much greater diversity, and thus effective ranking for generalist policies (Figure 4).** Additionally, we find that our task-aware ranking approach leads to more accurate rankings than both standard Elo computation and the conventional Bradley-Terry model. The resulting rankings of both, oracle and our approach follow the intuitions of prior work: expressive action representations outperform simple binning tokenization [5, 68, 41, 69], and discrete action tokenization ("FAST", "FSQ") outperform diffusion policies ("Flow", "π_0-DROID") in language-conditioned evaluations [5, 70] (Figure 3).

We also test using an average of the progress scores for each of the pairwise policy comparisons to compute a global ranking ("PROG"). The results in Figure 4 show that this strategy achieves good correlation to the oracle as well. However, ranking policies based on progress scores alone can miss

more nuanced feedback on policy performance. We find that evaluators regularly score both policies in an A/B comparison with the *same* progress score, yet express clear *preference* for one policy over the other, e.g., because it acted more swiftly or confidently (see Appendix E for examples). Thus, progress-based and preference-based rankings are complementary, and should both be reported.

4.3 RoboArena Evaluation is Sample Efficient

We investigate how many evaluations are required to produce an accurate ranking using RoboArena: we compute rankings for differently-sized, random subsets of our full evaluation data and report ranking accuracy as a function of the number of evaluation episodes in Figure 5. We find that RoboArena converges to high-quality rankings within just 100 pairwise comparisons, matching the convergence speed of conventional robot evaluations, while providing significantly more accurate rankings. The quality of RoboArena rankings further improves as more comparisons are collected. This suggests, that distributed RoboArena evaluations offer an appealing alternative to regular policy evaluations.

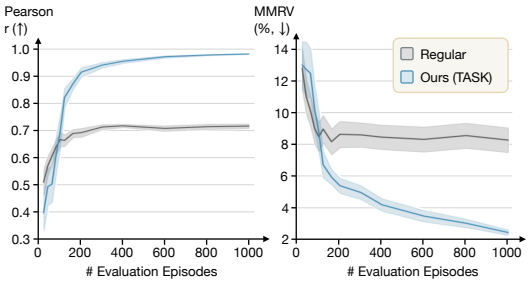

Figure 5: Rank correlation as a function of number of evaluation episodes. RoboArena converges to high-quality rankings within just 100 pairwise comparisons.

4.4 Extracting qualitative policy characteristics

In this section, we evaluate the quality of the LLM/VLM-assisted *policy reports* (Section 3.3). First, we evaluate the VLM category predictions by comparing them to category assignments made by a human expert, and we find that they are approximately 95% accurate (Figure 6).

Model	Task Category Acc.
GPT-4o	94.6%

Figure 6: Task categorization accuracy (448 samples).

Next, we examine whether the comparative claims in the generated policy report align with the evaluation data, using the example of the strongest policy in our pool: π_0-FAST-DROID [5]. Concretely, the report compares π_0-FAST-DROID's performance to that of other policies in the pool along different task categories. In Figure 7 we show, that for most categories for which the report claims that π_0-FAST-DROID outperforms, matches, or underperforms other policies, these claims are supported by the respective win rates in the evaluation data. We encourage readers to review the full, interactive report with video references in our supplementary material.

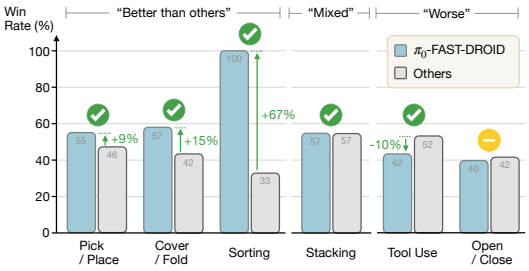

Figure 7: Claims made by our LLM-assisted analysis tools about π_0-FAST-DROID outperforming, matching, or underperforming other policies are supported by the win rates in the evaluation data.

5 Discussion

We introduced RoboArena, a distributed framework for evaluating generalist robot policies. We have shown that by aggregating evaluations across a decentralized network of evaluators, each running pairwise policy comparisons on many different tasks and scenes, RoboArena can generate more accurate policy performance rankings than conventional, centralized evaluation approaches, while retaining high evaluation sample efficiency. We have also introduced prototype tools for LLM-assisted analysis of the evaluation results. We will open-source our RoboArena evaluation framework and give other researchers access for contributing policies and evaluations, in an effort to make evaluations of generalist robot policies more comparable.

6 Limitations

Cross-embodiment. While RoboArena is a general approach for robot evaluation, the experimental evaluations in this paper have focused on the DROID platform [1], since it was well-suited for developing our evaluation framework. However, there is a growing interest in developing *cross-embodiment* policies, that can not only operate across many scenes and tasks, but also across robot embodiments. Future work should investigate how RoboArena can be extended to diverse robot embodiments and still support policies that may only be evaluatable on specific embodiments.

Controlled experimentation. The design of RoboArena, which is focused on decentralized evaluation without restrictions on tasks or scenes, makes it challenging to perform experiments that only vary *a single* condition at a time (e.g. only camera angle, or only object position). As such, RoboArena is complementary to targeted, smaller-scale evaluations that focus on individual axes of generalization [71].

Adversarial evaluators. While RoboArena's distributed, double-blind evaluation scheme gives it an inherent robustness against individual influencing, we have not investigated its robustness to intentionally adversarial evaluators that try to temper with evaluation results, for example by providing random preference ratings or intentionally misleading language feedback. Future work should investigate how distributed robot evaluation approaches can be hardened against such tampering.

Over-optimization and Gotthart's Law. A common wisdom is that a measure, e.g., for model performance, ceases to be a good measure when it becomes a target. This potential for over-optimization, also known as Gotthart's law, is innate to any measure, and has recently been shown to also impact crowd-sourced evaluations [72], even if they may inherently be more robust to such over-optimization than static benchmarks. While in robotics, the current (limited) performance of policies makes such over-optimization less likely, future work should critically examine whether evaluations with approaches like RoboArena remain well-correlated with perceived real-world policy performance.

Acknowledgments

We thank Siyi Huang, Ellie Huynh, Emiliano Adrian Sanchez, Justin Kang, and Sarah Kunda for help with evaluating policies. We also thank Kyle Stachowicz for help with repairing the DROID robot station during development. This research was partly supported by RAI, ONR N00014-25-1-2060, ONR N00014-22-1-2621, ONR N00014-22-12677NSF, NFS IIS-2150826, NSF CAREER 2239301, NSF 2331783, DARPA HR00112490428, DARPA TIAMAT HR00112490421, the Amazon Science Hub, the Toyota Research Institute, the Army Research Lab, and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (RS-2024-00333634). We also want to acknowledge funding support from Natural Sciences and Engineering Research Council of Canada, Fonds de recherche du Québec and The Canadian Institute for Advanced Research (CIFAR) and compute support from Digital Research Alliance of Canada, Mila IDT, and NVidia. We would also like to thank John Edwards Leadership Fund (CFI) for funding the purchase of part of the hardware for this project.

References

[1] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang,

P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024.

[2] Open X-Embodiment 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.

[3] 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.

[4] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. M. Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. *arXiv preprint arXiv:2409.05865*, 2024.

[5] 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. *Robotics: Science and Systems*, 2025.

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

[7] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[8] B. Yang, D. Jayaraman, J. Zhang, and S. Levine. Replab: A reproducible low-cost arm benchmark for robotic learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8691–8697. IEEE, 2019.

[9] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[10] 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. https://functional-manipulation-benchmark.github.io, 2023.

[11] 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.

[12] 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.

[13] G. Zhou, V. Dean, M. K. Srirama, A. Rajeswaran, J. Pari, K. Hatch, A. Jain, T. Yu, P. Abbeel, L. Pinto, C. Finn, and A. Gupta. Train offline, test online: A real robot learning benchmark, 2023.

[14] S. Yenamandra, A. Ramachandran, M. Khanna, K. Yadav, J. Vakil, A. Melnik, M. Büttner, L. Harz, L. Brown, G. C. Nandi, et al. Towards open-world mobile manipulation in homes: Lessons from the neurips 2023 homerobot open vocabulary mobile manipulation challenge. *arXiv preprint arXiv:2407.06939*, 2024.

[15] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M. Jordan, J. E. Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

[16] D. Jiang, M. Ku, T. Li, Y. Ni, S. Sun, R. Fan, and W. Chen. Genai arena: An open evaluation platform for generative models. *Advances in Neural Information Processing Systems*, 37: 79889–79908, 2024.

[17] 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.

[18] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. 2012.

[19] 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.

[20] 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.

[21] Y. Lee, E. S. Hu, Z. Yang, A. Yin, and J. J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. *ICRA*, 2021. URL https://clvrai.com/furniture.

[22] 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.

[23] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR, 2022.

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

[25] C. Bao, H. Xu, Y. Qin, and X. Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.

[26] Y. Xiang, X. Wang, S. Hu, B. Zhu, X. Huang, X. Wu, and S. Lyu. Rmbench: Benchmarking deep reinforcement learning for robotic manipulator control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1207–1214. IEEE, 2023.

[27] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

[28] 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.

[29] 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.

[30] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. *arXiv preprint arXiv:2503.24278*, 2025.

[31] J. Collins, M. Robson, J. Yamada, M. Sridharan, K. Janik, and I. Posner. Ramp: A benchmark for evaluating robotic assembly manipulation and planning. *IEEE Robotics and Automation Letters*, 9(1):9–16, 2023.

[32] N. Khargonkar, S. H. Allu, Y. Lu, B. Prabhakaran, Y. Xiang, et al. Scenereplica: Benchmarking real-world robot manipulation by creating replicable scenes. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8258–8264. IEEE, 2024.

[33] 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.

[34] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for ai. *AI magazine*, 18(1):73–73, 1997.

[35] 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.

[36] The Earth Rover Challenge Organizers. The Earth Rover Challenge. https://sites.google.com/view/the-earth-rover-challenge/, 2025.

[37] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[38] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

[39] 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, 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.

[40] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference on Robot Learning*, 2024.

[41] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

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

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

[45] A. Szot, B. Mazoure, O. Attia, A. Timofeev, H. Agrawal, D. Hjelm, Z. Gan, Z. Kira, and A. Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. *arXiv preprint arXiv:2412.08442*, 2024.

[46] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

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

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

[49] 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.

[50] 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.

[51] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4788–4795. IEEE, 2024.

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

[53] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[54] AgiBot-World-Contributors, Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Hu, X. Huang, S. Jiang, Y. Jiang, C. Jing, H. Li, J. Li, C. Liu, Y. Liu, Y. Lu, J. Luo, P. Luo, Y. Mu, Y. Niu, Y. Pan, J. Pang, Y. Qiao, G. Ren, C. Ruan, J. Shan, Y. Shen, C. Shi, M. Shi, M. Shi, C. Sima, J. Song, H. Wang, W. Wang, D. Wei, C. Xie, G. Xu, J. Yan, C. Yang, L. Yang, S. Yang, M. Yao, J. Zeng, C. Zhang, Q. Zhang, B. Zhao, C. Zhao, J. Zhao, and J. Zhu. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[55] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.

[56] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[57] N. Guha, J. Nyarko, D. Ho, C. Ré, A. Chilton, A. Chohlas-Wood, A. Peters, B. Waldon, D. Rockmore, D. Zambrano, et al. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Advances in Neural Information Processing Systems*, 36:44123–44279, 2023.

[58] S. Dasari, J. Wang, J. Hong, S. Bahl, Y. Lin, A. Wang, A. Thankaraj, K. Chahal, B. Calli, S. Gupta, et al. Rb2: Robotic manipulation benchmarking with a twist. *arXiv preprint arXiv:2203.08098*, 2022.

[59] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *arXiv preprint arXiv:2409.09491*, 2024.

[60] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[61] A. E. Elo. The proposed uscf rating system, its development, theory, and applications. *Chess life*, 22(8):242–247, 1967.

[62] E. Zermelo. Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29(1):436–460, 1929.

[63] R. R. Davidson. On extending the bradley-terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association*, 65(329):317–328, 1970.

[64] 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.

[65] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen. Finite scalar quantization: Vq-vae made simple, 2023. URL https://arxiv.org/abs/2309.15505.

[66] P. Intelligence. Openpi. https://github.com/Physical-Intelligence/openpi, 2025.

[67] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, 2024.

[68] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[69] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[70] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[71] 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.

[72] The Register. Regarding Meta, Llama 4, and Cheating, 2025. URL https://www.theregister.com/2025/04/08/meta_llama4_cheating/.

[73] A. Z. Ren. Open pi zero: An open source raspberry pi zero. https://github.com/allenzren/open-pi-zero, 2025.

A Contributions

RoboArena system design and implementation: Pranav Atreya, Karl Pertsch, Tony Lee

Experiment design and analysis: Pranav Atreya, Karl Pertsch, Tony Lee

Policy evaluation: Pranav Atreya, Tony Lee, Moo Jin Kim, Karl Pertsch, Arhan Jain, Artur Kuramshin, Cyrus Neary, Edward Hu, Kanav Arora, Kirsty Ellis, Luca Macesanu, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Jie Wang, Xingfang Yuan

Simulated evaluation environments: Arhan Jain, Karl Pertsch, Xuning Yang, Clemens Eppner, Fabio Ramos, Jonathan Tremblay

Paper writing: Karl Pertsch, Pranav Atreya, Tony Lee

Project coordination: Karl Pertsch, Pranav Atreya

Advising: Sergey Levine, Chelsea Finn, Percy Liang, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martin-Martin, Youngwoon Lee

B Policy Ranking EM Procedure

Here we give additional details on our proposed policy ranking algorithm, introduced in section 3.2. We first describe the underlying probabilistic model, derive all necessary gradients and Hessians, then present the complete EM pseudocode. We then show how the algorithm can be easily modified to also use partial-success information when present, outline the algorithms behind the other ranking procedures discussed in the paper, and then list all hyperparameters.

B.1 Model Definition

Observed data. We assume a fixed set of N policies

$$\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}.$$

Evaluators compare policies in A/B trials: in trial n, they pit policy π_{i_n} against π_{j_n} on some (latent) task and report an outcome

$$y_n \in \{0, 1, 2\}, \quad 0 = \text{loss}, \ 1 = \text{tie}, \ 2 = \text{win}, \quad n = 1, \ldots, M.$$

(Note this is a generalization of the model described in section 3.2, which did not allow for ties.) We collect all M observations into

$$\mathcal{D}_p = \left\{ \left(i_n, j_n, y_n\right) \right\}_{n=1}^{M}.$$

At no point do we assume the true "task" identifiers are observed; instead we infer a small number of latent "task-buckets" that capture varying difficulty levels.

Latent tasks and mixture weights. To model varying evaluation conditions, we assume each trial belongs—unobserved—to one of T difficulty categories, or "buckets." We place a discrete prior over these buckets,

$$\nu = (\nu_1, \ldots, \nu_T), \quad \nu_t \geq 0, \ \sum_{t=1}^{T} \nu_t = 1,$$

so that before seeing any data, the probability a random trial uses bucket t is ν_t.

Per-policy and per-bucket parameters. Each policy π_p has a global "log-ability" parameter θ_p. Additionally, each policy may perform better or worse on certain buckets, so we introduce an offset $\psi_{p,t}$ for policy p on bucket t. This offset serves the purpose of enabling two policies to perform differently relative to each other on different tasks (the motivation for the introduction of this offset is discussed further in section 3.2). Finally, each bucket t has a base difficulty τ_t. Collecting these,

$$\theta = (\theta_1, \ldots, \theta_N), \quad \psi = \{\psi_{p,t}\}_{p=1..N,\ t=1..T}, \quad \tau = (\tau_1, \ldots, \tau_T).$$

Intuitively, θ_p captures overall policy strength, τ_t captures bucket difficulty, and $\psi_{p,t}$ models policy-specific ease or difficulty adjustments within each bucket.

Tie-model parameter. We allow ties via the Davidson extension, introducing a single scalar $\nu_{\text{tie}} \in (0, 1)$ which scales the draw probability relative to wins and losses.

Link function and conditional likelihood. Given trial n and hypothesized bucket t, compute each policy's log-odds,

$$z_{i_n,t} = \theta_{i_n} + \psi_{i_n,t} - \tau_t, \quad z_{j_n,t} = \theta_{j_n} + \psi_{j_n,t} - \tau_t.$$

Pass these through the logistic sigmoid (our instantiation of the *link-function*, which generally is a mapping from linear predictors, here our log-odds, to probabilities) $\sigma(z) = 1/(1 + e^{-z})$ to get

$$q_{i_n,t} = \sigma(z_{i_n,t}), \quad q_{j_n,t} = \sigma(z_{j_n,t}).$$

Under the "independent-solve" assumption, policy i_n "solves" the task with probability $q_{i_n,t}$ and j_n with probability $q_{j_n,t}$, independently given t. Even though our dataset represents preferences between policies i and j, this "independent-solve" assumption is natural because in actuality, the performance of policy i on the task is independent to that of policy j. Thus

$$P(y_n = 2 \mid t) = q_{i_n,t}\,\big(1 - q_{j_n,t}\big), \quad P(y_n = 0 \mid t) = \big(1 - q_{i_n,t}\big)\,q_{j_n,t},$$

$$P(y_n = 1 \mid t) = 2\,\nu_{\text{tie}}\,\sqrt{q_{i_n,t}(1 - q_{i_n,t})\,q_{j_n,t}(1 - q_{j_n,t})}\,,$$

again invoking the Davidson extension of the Bradley-Terry model. Because t is latent, we marginalize over buckets:

$$P(y_n) = \sum_{t=1}^{T} \nu_t\; P\big(y_n \mid t\big).$$

This completes the model definition.

B.2 Gradient & Hessian Derivation

Below we derive all first and second derivatives needed for the clipped-Newton M-step. We begin by writing the expected complete-data objective and then expand gradients and Hessians for each parameter block.

Expected complete-data objective. Denote the full parameter set by $\Theta = (\theta, \psi, \tau, \nu, \nu_{\text{tie}})$. In the E–step we compute responsibilities

$$\gamma_{n,t} = P\big(t \mid y_n; \Theta^{\text{old}}\big) = \frac{\nu_t\, P(y_n \mid t; \Theta^{\text{old}})}{\sum_{u=1}^{T} \nu_u\, P(y_n \mid u; \Theta^{\text{old}})}.$$

The expected complete-data log-likelihood, including L2 penalties $\lambda_\theta, \lambda_\psi$, is

$$Q(\Theta) = \sum_{n=1}^{M}\sum_{t=1}^{T} \gamma_{n,t}\,\Big[\log \nu_t + \log P(y_n \mid t)\Big] \;-\; \frac{\lambda_\theta}{2}\sum_{p=1}^{N}\theta_p^2 \;-\; \frac{\lambda_\psi}{2}\sum_{p=1}^{N}\sum_{t=1}^{T}\psi_{p,t}^2.$$

Gradient w.r.t. θ_p. Only trials where policy p appears contribute. Let $\mathcal{I}_p = \{n : i_n = p\}$, $\mathcal{J}_p = \{n : j_n = p\}$. Then

$$\frac{\partial Q}{\partial \theta_p} = \sum_{n \in \mathcal{I}_p}\sum_{t=1}^{T} \gamma_{n,t}\,\frac{\partial}{\partial z_{i_n,t}} \log P(y_n \mid t) - \sum_{n \in \mathcal{J}_p}\sum_{t=1}^{T} \gamma_{n,t}\,\frac{\partial}{\partial z_{j_n,t}} \log P(y_n \mid t) - \lambda_\theta\,\theta_p.$$

For instance, for $y_n = 2$ (a "win"), one finds

$$\frac{\partial}{\partial z_{i,t}} \log P(y_n = 2 \mid t) = 1 - \sigma(z_{i,t}), \qquad \frac{\partial}{\partial z_{j,t}} \log P(y_n = 2 \mid t) = -\,\sigma(z_{j,t}),$$

with analogous expressions in the loss or tie case.

Hessian w.r.t. θ_p. The second derivative accumulates the negative of the logistic variances:

$$\frac{\partial^2 Q}{\partial \theta_p^2} = -\sum_{n \in \mathcal{I}_p} \sum_t \gamma_{n,t}\, \sigma(z_{i,t})(1-\sigma(z_{i,t})) - \sum_{n \in \mathcal{J}_p} \sum_t \gamma_{n,t}\, \sigma(z_{j,t})(1-\sigma(z_{j,t})) - \lambda_\theta.$$

Gradients/Hessians for $\psi_{p,t}$. Each $\psi_{p,t}$ enters exactly like θ_p but only for bucket t:

$$\frac{\partial Q}{\partial \psi_{p,t}} = \sum_{n: i_n = p} \gamma_{n,t}\, \frac{\partial \log P}{\partial z_{i_n,t}} - \sum_{n: j_n = p} \gamma_{n,t}\, \frac{\partial \log P}{\partial z_{j_n,t}} - \lambda_\psi\, \psi_{p,t},$$

$$\frac{\partial^2 Q}{\partial \psi_{p,t}^2} = -\sum_{n: i_n = p} \gamma_{n,t}\, \sigma(z_{i,t})(1-\sigma(z_{i,t})) - \sum_{n: j_n = p} \gamma_{n,t}\, \sigma(z_{j,t})(1-\sigma(z_{j,t})) - \lambda_\psi.$$

Gradients/Hessians for τ_t. Since τ_t enters with a minus sign in both i and j,

$$\frac{\partial Q}{\partial \tau_t} = -\sum_{n=1}^{M} \gamma_{n,t} \left[\tfrac{\partial}{\partial z_{i,t}} \log P(y_n \mid t) + \tfrac{\partial}{\partial z_{j,t}} \log P(y_n \mid t) \right],$$

$$\frac{\partial^2 Q}{\partial \tau_t^2} = -\sum_{n=1}^{M} \gamma_{n,t} \left[\sigma(z_{i,t})(1-\sigma(z_{i,t})) + \sigma(z_{j,t})(1-\sigma(z_{j,t})) \right].$$

All of these gradient and Hessian terms are used in the M-step clipped-Newton updates described in Algorithm 2.

B.3 Including Partial-Success Information

When available, the algorithm can be easily modified to make use of partial success information $s_n^{(i)}, s_n^{(j)} \in [0,1]$. We introduce an extra Gaussian term $\exp[-((s^{(i)} - q_i)^2 + (s^{(j)} - q_j)^2)/(2\sigma^2_{\text{partial}})]^{w_{\text{ps}}}$ in the E–step likelihood and add its gradients/Hessians in the M–step exactly as in sections B.1 and B.2. The modified pseudocode adds two lines in the E–step and augments each g, h with the partial-success contributions.

B.4 Baseline Ranking Algorithms

Below we summarize the simpler ranking methods we compare against, including both classic preference-only approaches and a simple partial-success approach.

Bradley–Terry MLE (offline). In the standard Bradley–Terry maximum-likelihood estimator, we posit

$$P(y_n = 2) \;=\; \sigma\big(\theta_{i_n} - \theta_{j_n}\big),$$

and fit the ability parameters θ by maximizing the log-likelihood of all win/loss outcomes (ties can be handled via small modifications). A simple gradient-ascent update is

$$\theta_p \;\leftarrow\; \theta_p \;+\; \eta \sum_{n:\, i_n = p} \Big[y_n - \sigma\big(\theta_{i_n} - \theta_{j_n}\big) \Big] \;-\; \eta \sum_{n:\, j_n = p} \Big[y_n - \sigma\big(\theta_{i_n} - \theta_{j_n}\big) \Big],$$

where η is a small learning rate, $y_n = 2$ for a win by i_n, $y_n = 0$ for a loss, and ties are typically treated as half-win/half-loss. Iterating this update until convergence yields the offline MLE of the BT model. In practice one also adds L2 regularization $\lambda\theta_p$ to stabilize training.

Algorithm 2 EM for Fitting Model Parameters

Require: Data $\{(i_n, j_n, y_n)\}_{n=1}^{M}$, number of buckets T, max iters EM_ITERS
Ensure: $\theta,\ \psi,\ \tau,\ \nu,\ \nu_{\text{tie}}$ and sorted policy ranking

Initialize:
  $\theta_p \sim \mathcal{N}(0, 0.1)$ for $p = 1, \ldots, N$
  $\psi_{p,t} \leftarrow 0$ for all p, t
  $\tau_t \sim \mathcal{N}(0, 0.1)$ for $t = 1, \ldots, T$
  Mixture weights: $\nu_t \leftarrow 1/T$ for $t = 1, \ldots, T$
  Tie-parameter: $\nu_{\text{tie}} \leftarrow 0.5$
for $m = 1$ **to** EM_ITERS **do**
  ▷ — E–step: compute responsibilities —
    for $n = 1$ **to** M **do**
      compute $P(y_n \mid t)$ for each bucket t
      $\gamma_{n,t} \leftarrow \nu_t\, P(y_n \mid t)$
      normalize $\{\gamma_{n,1}, \ldots, \gamma_{n,T}\}$ so they sum to 1
  ▷ — M–step: clipped-Newton updates —
    for each policy $p = 1, \ldots, N$ **do**
      compute gradient $g_p = \partial Q / \partial \theta_p$ and Hessian $h_p = \partial^2 Q / \partial \theta_p^2$
      $\theta_p \leftarrow \theta_p - \text{clip}\big(g_p / h_p\big)$
    for each (p, t) with $p = 1..N,\ t = 1..T$ **do**
      compute $g_{p,t} = \partial Q / \partial \psi_{p,t}$ and $h_{p,t} = \partial^2 Q / \partial \psi_{p,t}^2$
      $\psi_{p,t} \leftarrow \psi_{p,t} - \text{clip}\big(g_{p,t} / h_{p,t}\big)$
    for each bucket $t = 1, \ldots, T$ **do**
      compute $g_t = \partial Q / \partial \tau_t$ and $h_t = \partial^2 Q / \partial \tau_t^2$
      $\tau_t \leftarrow \tau_t - \text{clip}\big(g_t / h_t\big)$
    Update mixture weights and tie-rate:
      $\nu_t \leftarrow \dfrac{1}{M} \sum_{n=1}^{M} \gamma_{n,t}$
      $\nu_{\text{tie}} \leftarrow 0.5 \times \dfrac{\sum_{n,t} \gamma_{n,t}\, p_{\text{tie}}(n,t)}{\sum_{n,t} \gamma_{n,t}\, p_{\text{win}}(n,t)}$
    if maximum change in any $\theta_p <$ tol **then**
      break
return policies sorted in descending order of θ_p

Elo-style online update. Elo is a one-pass, online version of BT often used in rating chess players. After each observed comparison between A and B, one updates only the two involved ratings:

$$\theta_A \ \leftarrow\ \theta_A \ +\ K\,\Big(y_{AB} - \sigma(\theta_A - \theta_B)\Big), \quad \theta_B \ \leftarrow\ \theta_B \ -\ K\,\Big(y_{AB} - \sigma(\theta_A - \theta_B)\Big),$$

where K is a constant "K-factor," and $y_{AB} = 1$ if A wins, 0 if A loses (ties set $y_{AB} = 0.5$). This update has the property of immediate rating adjustments as data arrives and requires no global passes over the dataset.

Partial-Success Averaging. As a non-preference baseline, one can ignore all binary outcomes and instead rank policies by their *average partial-success rate* across all rollouts. Concretely, if policy p receives a fractional success score $s_n^{(p)} \in [0, 1]$ on each trial n, we compute

$$\bar{s}_p \ =\ \frac{1}{N_p} \sum_{n:\, i_n = p \text{ or } j_n = p} \big[s_n^{(p)}\big],$$

where N_p is the total number of rollouts involving p. Finally, we sort policies in descending order of $\bar{s}_p$. This method leverages continuous performance feedback directly, but does not account for the paired-comparison structure of A/B evaluations.

B.5 Hyperparameter Table

Parameter	Default	Search range
EM_ITERS	60	$\{50, 100, 200\}$
# buckets T	60	$\{10...200\}$
step_clip	1.0	$[0.1,\ 10]$
l2_theta	10^{-2}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
l2_psi	10^{-2}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
step_decay	0.99	$\{0.9, 0.99, 0.999\}$
tol	10^{-4}	$\{10^{-5}, 10^{-4}\}$

Table 1: EM procedure hyperparameters.

B.6 Simulating Shifts in the Task Distribution and Policy Set

The intent of the RoboArena platform is for it to serve as a long-running evaluation and benchmarking resource for the robotics community. In such a setting, it is likely that the policy set will evolve over time, with weaker policies getting dropped and stronger policies getting added to the pool. At the same time, as the capabilities of policies evolve, evaluators will likely adjust the tasks they set up for evaluation to better test the new policy capabilities.

We test whether the RoboArena ranking model will remain valid and performant under distribution shifts over time. We rerun our rankings with artificial drift in task difficulty (as judged by average per-task progress scores), starting with easy tasks and moving towards harder tasks over time, while at the same time phasing out weaker policies and adding stronger policies to the evaluation pool over time (again judged by per-policy progress scores). This simulates RoboArena's development as a community benchmark with increasingly capable policies and increasingly challenging task distributions. The results in Table 2 show that even under these more challenging circumstances, RoboArena evaluations retain significantly higher correlation with oracle scores than "Regular" evaluations in a single laboratory.

Eval Approach	**Pearson** $r(\uparrow)$	**MMRV** $(\downarrow)$
Regular	0.692	0.141
RoboArena w/ dist. shifts (ours)	**0.838**	**0.058**

Table 2: RoboArena under simulated distribution shifts.

C The DROID-RoboArena Evaluation System

We instantiate RoboArena in a prototype evaluation system on the DROID robot platform [1], which consists of a Franka Panda 7DoF robot arm, a Robotiq 2F-85 parallel-jaw gripper, a ZED-mini stereo wrist camera and one or multiple external ZED 2 stereo cameras (see Figure 8). We choose the DROID platform since it offers multiple attractive properties:

- The Panda arm provides sufficient dexterity to perform a wide range of manipulation tasks across varied real-world environments.
- The robot is mounted on a height-adjustable, mobile table, which enables rapid reconfiguration of scenes and camera viewpoints.
- Most importantly, the platform is associated with the open-source DROID dataset [1], a large-scale, real-world dataset that supports training of *generalizable* robot policies.
- Finally, DROID setups are already deployed across multiple academic institutions worldwide, which is key to enabling distributed evaluation at scale.

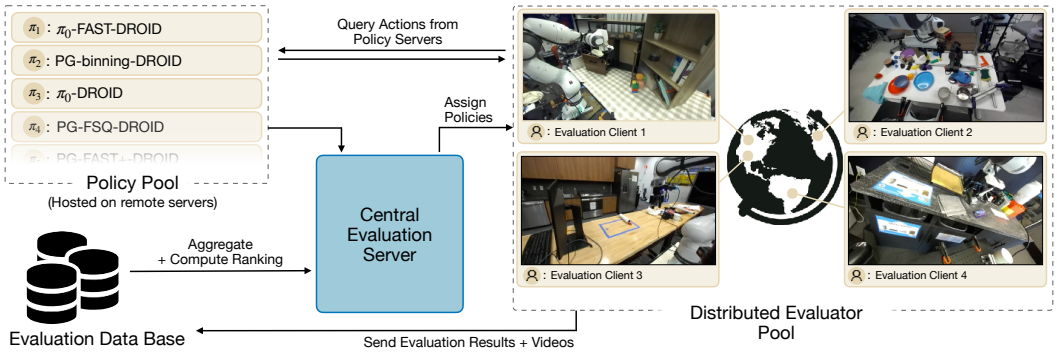

Figure 9: The DROID-RoboArena system consists of a pool of remotely hosted policy servers, a pool of distributed evaluator "clients" with real robot setups, a database for storing evaluation results, and a central evaluation management server that orchestrates communication, aggregates the evaluation results, and computes a policy ranking.

We note that, while DROID is a convenient platform to develop a prototype distributed robot evaluation, the RoboArena evaluation approach readily extends to other robot embodiments and potentially even to evaluating cross-embodiment policies [2] across platforms. The remainder of this section, we provide details on our evaluation system design (Appendix C.1) and outline safety considerations and incentive mechanisms that support open participation from the broader robotics community (Appendix C.2).

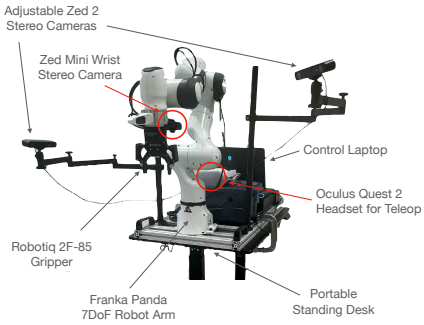

Figure 8: The DROID robot setup, which we use for the DROID-RoboArena evaluation system. Reproduced with permission from Khazatsky et al. [1].

C.1 RoboArena System Design

We design a system for performing decentralized evaluation over a potentially large pool of policies, with a pool of evaluators that may span many different locations, countries, or even continents. Our system has four core components: policy inference servers, evaluation clients, an evaluation database, and a central evaluation server (see Figure 9).

Policy inference servers We host all policies in our pool on remote servers, instead of running them "on-premise" at the robot evaluation station. This has multiple benefits: *effective resource utilization* since multiple evaluators can share the same policy server, a *lightweight evaluation client* since no inference compute needs to be provided client-side, which lowers the barrier for contributing evaluations, and *user-side control* over policy servers since we assume users host their own policies when submitting them for evaluation, which ensures that policies are run correctly, equipped with sufficient inference compute, and proprietary models can be evaluated without sharing model weights. Remote hosting may, however, introduce additional latency during policy inference, but in practice we found this to be negligible for the static manipulation tasks on which we typically evaluate DROID policies. We acknowledge that future versions of RoboArena evaluations may need to support (at least partially) local inference to enable evaluation of more dynamic tasks.

Evaluation clients We provide a client-side script that guides users through the evaluation protocol outlined in Section 3.1. It handles communication with the central evaluation server and the policy inference servers. Since no client-side inference compute is required, this script can be run on any computer that is connected to the Internet and the DROID robot.

Evaluation database This database hosts all evaluation results: instructions, scores and preference, natural language feedback, and the rollout videos. Information is uploaded by the clients.

Central evaluation server We host a central evaluation server that assigns policies to evaluators, and keeps track of newly registered or deprecated policies in the policy pool and automatically cancels evaluations after a timeout, e.g., if evaluation clients crashed or lost network connection.

C.2 Open-Sourcing RoboArena: Interfaces, Safety and Incentives

Our goal in designing the DROID-RoboArena is to provide it as a resource to the robotics community, to which anyone can submit policies and contribute evaluations. Importantly, DROID-RoboArena can enable researchers without access to a physical robot to train and evaluate real-world generalist robot policies – now all required resources are open-source: diverse, real-robot datasets [50, 2, 1, 54], scalable modeling frameworks [39, 3, 73, 66, 47], and DROID-RoboArena real-robot evaluations.

Making the DROID-RoboArena easy to use, safe, and self-sustaining requires a few additional elements. First, we make it easy to browse all performed evaluations and view existing policy leaderboards on our website. We plan to publish separate leaderboards for "all policies" and "open-source policies"; the latter are trained only on publicly available datasets, and provide weight access and details for how to reproduce training. Additionally, we design multiple **safety layers** to prevent policies from damaging robot evaluation hardware: first, we test that any newly submitted policy server complies with the expected input and output formats. Then we run said policy in a "test environment" across a few different scenes and tasks with a specifically trained evaluator that is able to quickly intervene if a policy runs the danger of acting unsafely and damaging the robot (akin to a test driver for autonomous vehicles). Only after a policy passed these tests, we add it to the general pool and distribute it to evaluators.

Finally, to make the DROID-RoboArena self-sufficient in the long term, we implement an **"evaluation credit" system**, that balances evaluation supply and demand: for every pairwise policy evaluation that an evaluator runs, they receive a credit, which they can use to request an equal number of pairwise comparisons between their own policies and other policies from the pool. This way, evaluation effort for any individual is comparable to running evaluations for just their own policy on their own setup, but by agreeing to evaluate others' policies through the decentralized benchmarking system, they effectively get a much broader evaluation coverage for the same effort. To support participation of researchers without access to a physical robot (and thus without the ability to contribute evaluations themselves), multiple institutions agreed to "sponsor" a base budget of evaluations that will be distributed among all users "free of charge".

D Evaluation Data Breakdown

Here we depict in detail the evaluation data collected by RoboArena, emphasizing it's diversity and scale. To the best of our knowledge, the evaluations done thus far with RoboArena constitute the most comprehensive evaluations of generalist policies to date.

Figure 10 depicts on the top the diversity in the *verb* making up the task command, and on the bottom the wide range of object classes that are being interacted with during the evaluation episodes. Generalist policies are uniquely data demanding when it comes to evaluations, as to properly assess a policy's performance as a generalist, the policy must be queried with a diverse array of tasks. Indeed, figure 10 shows that the RoboArena evaluation procedure permits the required diversity in task commands and objects. Figure 11 similarly shows a representation of the diversity of RoboArena evaluations with respect to the set of scenes upon which evaluations were performed. The environments sampled were chosen randomly from the pool of all evaluation episodes.

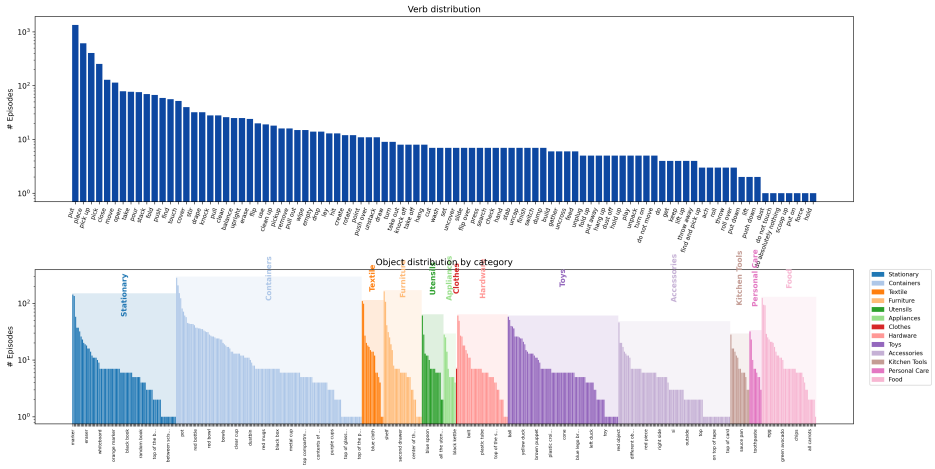

Figure 10: **Top:** bar chart of the most common verbs used in the task commands, along with their frequencies in the evaluation data collected. The task commands exhibit a diverse array of instructions, including ”uncapping”, ”unplugging”, ”finding”, ”dusting”, etc. **Bottom:** visualization of the diversity of object categories being commanded to be interacted with, bucketed by major objects types.

Figure 11: Environments in RoboArena are diverse, due to RoboArena's distributed nature and the eschewing of a standardization of tasks. Here we depict 32 sample environments used for RoboArena across the network of participating institutions. Evaluators were encouraged to scale diversity, leading to a visible heterogeneity of environments. Even when the physical location of the robot was the same, lighting, camera viewpoints, tablecloths, and objects were often altered significantly, again made easy by the fact that exact scenes and tasks need not be standardized.

E Nuances in Policy Preference Labels

Evaluators participating in the RoboArena evaluation framework are asked to provide a rough estimate of partial success of each rollout on the prescribed task, but also critically, a preference label specifying whether they liked policy A versus B on the same task. As we will outline here, this preference feedback can contain quite a bit of information beyond what is present in the partial success feedback. Concretely, figure 3 depicts a few examples where *partial success feedback was the same for policies A and B*, yet the evaluator marked a clear preference for either A or B.

We further found experimentally that for **11%** of all A/B evaluations, the preference feedback did not agree with the partial success feedback, either because partial feedback was equal but the preference was not, or partial feedback and preference feedback displayed opposing trends. Thus, preference feedback is a uniquely rich data source for policy evaluations.

Table 3: Examples of preference feedback for A/B pairs for which partial success feedback was equal for A and B, illustrating information beyond binary partial-success scores.

Session ID	Preferred Policy	Long-form Feedback
748	A	Although both policies don't put the cloth on the screwdriver, policy A places the cloth close to the banana, while policy B does not seem close to either object.
674	B	Both policies completed the entire task but policy B did it on the first try. After the first grasp and lift, it feels like policy A dropped the mouse prematurely. It then picked it up again and moved it further.
660	B	Policy B initially hesitated to move long distance but later transitioned to effective and rapid movements. Meanwhile, policy A also succeeded at the task, but it exhibited more sluggish movements.
649	A	Both A and B picked up the cup instead of pushing, and both then placed it on the table. After letting go, A returned to a starting pose while B kept repeatedly grabbing the cup, which is sub-optimal.
648	B	While both did take the object out of the bowl (qualifying for full score), B placed the object on the table area next to the bowl. This is the more natural thing to do.
641	A	Both policy A and policy B almost solved the task completely. However, policy A displayed more decisive motions with less corrective behavior while policy B solved the task by chance after multiple attempts.
579	A	Both policies were able to close the drawer most of the way. However, policy A went straight for the drawer and didn't stop until it was closed. Policy B got distracted by the plastic food items halfway through (though it eventually returned to close the drawer).

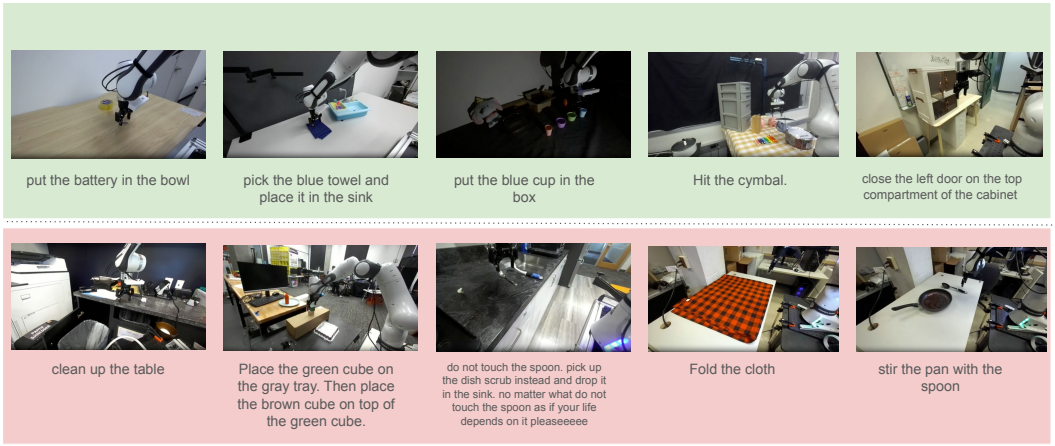

Figure 12: We observe that policies tend to succeed on tasks involving direct object manipulation, such as placing objects into containers (examples in green), but often fail in tasks that require tool use or nuanced semantic understanding, such as wiping, cleaning or following detailed multi-step instructions (examples in red).

F Policy Performance Reports

To assist in the analysis, we use our automated pipeline to generate detailed reports that summarize the behavior and performance of each policy across a wide range of manipulation tasks based on data collected during distributed evaluation episodes. In the pipeline, we prompt OpenAI's `o3-2025-04-16` model with the prompt template shown in Figure 14 to automatically generate a structured policy performance report for a given policy. We include a representative full report for π_0-FAST-DROID in Figure 13 with the video references removed. We further con-

cisely summarize the resulting full report using the prompt in Figure 15. The reports for all evaluated policies *with video references to specific evaluation episodes* are available at https://robo-arena.github.io/leaderboard.

G Strengths and Weaknesses of the Evaluated Generalist Policies

To our knowledge, the evaluation done in this work is the most comprehensive evaluation of generalist policies to date. Analyzing the head-to-head comparisons across a diverse set of real-world tasks, we identify consistent behavioral patterns and failure modes exhibited by current policies. In this section, we synthesize the general strengths and weaknesses observed across all evaluated policies and the trends that emerge across policy families.

A primary strength of these generalist policies is their ability to operate in diverse viewpoints, lighting conditions, and background appearances. Across the evaluated policies, we observe that tasks involving direct object manipulation (e.g., pick-and-place, pushing, toppling, and simple open/close tasks) are more reliably solved than those requiring tool use, cloth manipulation or complex semantic understanding (Figure 12). In particular, these policies tend to perform better when goals are straightforward and visually grounded but struggle when the task demands precision alignment, multi-step reasoning or specific attribute perception like object class or color. Tasks involving deformable objects (e.g., folding, draping) and actions with tools (e.g., wiping, scooping) remain key challenges, often resulting in partial motion or outright frozen behavior. The weaknesses highlight the gaps in generalization, robustness, and physical interaction across various manipulation tasks.

When comparing the different policy families, autoregressive policies (e.g., PG-FAST-DROID, PG-FAST+-DROID, π_0-FAST-DROID) consistently achieve higher success rates in pick-and-place, stacking, and classification tasks, due to their more precise language following. Diffusion-based policies (e.g., PG-flow-DROID, π_0-flow-DROID) perform well in fluid or continuous motion tasks like sliding and wiping but often lag in tasks have precise language instructions (e.g., knocking over specific objects). Binning policies (e.g., PG-Bin-DROID) consistently underperform in nearly all tasks, with frequent inactivity and low task completion. Overall, substantial work remains to be done to achieve reliable and generalizable robot policies.

1. Policy Overview
pi0_fast_droid is a fast, reactive manipulation policy that rarely idles and generally succeeds at single-object pick-and-place routines. When the goal object is visually salient it plans a direct path, grasps confidently, and often retries after a failed grasp. The policy copes reasonably well with mild clutter and can manipulate flexible objects such as towels or cloths. Limitations become evident whenever the task demands fine tool control, accurate alignment (e.g., insertions or stacking), deliberate inaction, or multi-step reasoning. In those settings the controller may oscillate, freeze, or grasp the wrong item, and it occasionally terminates without releasing the object it is holding.

2. Comparative Performance (head-to-head)
• Pick and Place – Across dozens of episodes pi0_fast_droid beat or tied the competing policy more often than it lost. It routinely grasped the correct item and reached the target location, while rival policies either froze or mis-grasped (e.g., ducks into cups, cups into bowls, blocks into trays).
• Cover / Drape / Fold – The policy consistently outperformed its counterpart; it was usually the only agent to lift cloth or achieve partial folding, whereas competitors often merely poked at the fabric.
• Sorting / Classification – In repeated colour-based sorting tasks pi0_fast_droid identified target colours correctly and placed them, while the alternative policies hesitated or selected wrong colours.
• Move / Slide – When required to slide or reposition an object, the policy succeeded with smoother, faster trajectories; rival controllers tended to overshoot or stall.
• Tool Use – Performance lagged behind the comparison agent: pi0_fast_droid lost or tied in most erasing, wiping, stirring or "clean the table" episodes, whereas the other policies executed smoother tool contact and fewer redundant motions.
• Open / Close – The policy underperformed; it either failed to latch onto handles or to finish closing motions, while the competing agent completed the same drawer or cabinet tasks more reliably.
• Object Manipulation – When asked to re-orient blocks, pour or open bottles, the policy frequently lost or tied; the competitor could align objects more precisely or at least avoid freezing.
• Group / Organize / Stack – Stacking success was mixed; pi0_fast_droid often placed items near the correct spot but the rival policy achieved proper alignment more often, leading to several losses.

3. Strengths
• Robust grasping of familiar rigid objects; e.g., picked the cup and placed it into a basket smoothly, and removed a block from a box despite flaps in the way.
• Re-attempt behaviour: after missing the bowl the first time it re-planned, re-grasped, and completed the pineapple placement.
• Effective colour/shape recognition that supports sorting and non-red discrimination tasks.
• Cloth manipulation: successfully folded a towel and achieved full coverage over a piggy-bank while the opponent merely poked.
• Quick object search sweeps that cover shelves and table surfaces before freezing competitors, e.g., "find the creeper toy".

4. Weaknesses
• Frequent freezing or limited exploration after one failed attempt, stalling entire episodes.
• Tool control deficiencies: could not keep the eraser in contact with the board or open a water bottle despite grasping the cap.
• Mis-identification of targets in clutter, e.g., placed tape in the wrong plate and grasped the robot instead of the marker meant to hit it.
• Difficulty with precision insertions (portafilter into grinder, small cubes onto stacks) leading to losses.
• Does not always release objects after placement, leaving grasped items hovering and tasks incomplete.

5. Instruction Following
• Handles colour and spatial qualifiers well ("blue cup into yellow bowl" succeeded).
• Fails at "do absolutely nothing" – still moved despite the explicit negation.
• Interprets minor typos ("non-read" → non-red) correctly and complied.
• Multi-step or relational instructions are followed inconsistently; it dropped the towel but never folded it in "place carrot then fold towel".
• Occasionally ignores action verbs and grasps the wrong reference (picked the robot instead of using the marker to hit it).

6. Reasoning
Scene reasoning strengths: correctly inferred colour grouping goals and located all cups quickly in cluttered scenes.
Weaknesses: often violates order constraints (tried to stack before tray placement) or stops after partial completion (emptied only one item then froze). Spatial reasoning is sometimes coarse; the controller places objects "near" rather than "inside/on top", leading to almost-correct states that still lose.

7. Manipulation Skills
• Grasping: robust with medium-sized rigid objects (cups, blocks, towels).
• Placing: accurate onto large, open targets; less precise for narrow targets or stacking (frequent mis-alignment of tape rolls).
• Stacking/Inserting: partial success, but alignment errors common; blocks often dropped from height.
• Tool manipulation: weak torque control when erasing, wiping or turning caps; slips off tools or hovers without making contact.
• Recovery: will back-off and re-grasp after a miss instead of giving up.
• Release: occasionally forgets to open gripper after placement.

8. Robustness to Scene Variations
• Handles moderate clutter and distractors, succeeding in busy kitchen and office scenes.
• Cloth backgrounds, patterned tables, and partial occlusions rarely confuse its perception.
• Sensitive to low-light episodes: performance degraded in dim "Place cup right side up" and "Put the yellow ducks in mug" scenes.
• Wrist-camera occlusions sometimes cause it to mis-localise small targets (e.g., screwdriver into mug task).

9. Common Failure Modes
• Freezing after first error or mid-air hover with object still grasped.
• Grasping the correct item but never releasing it into the goal.
• Selecting a distractor of similar colour/shape (tape vs. stapler, carrot vs. duck).
• Over-shooting and colliding with cabinets or shelving.
• Tool tasks: pushes or nudges the tool instead of forming a stable grasp, leading to repeated but ineffective motions.
• Misinterpreting passive, negated, or multi-step instructions (moved during "do not move", folded towel step omitted, etc.).

Overall, pi0_fast_droid provides a solid baseline for everyday pick-and-place and cloth-handling tasks, but would benefit from improved tool manipulation control, tighter release logic, and more deliberate high-level planning for multi-step or precision-alignment scenarios.

Figure 13: The full generated policy performance report for π_0-FAST-DROID (video references removed; check https://robo-arena.github.io/leaderboard for interactive reports with references).

We are evaluating a policy named `POLICY NAME` deployed on a robot arm to perform various manipulation tasks.
This policy was compared head-to-head against other policies across multiple episodes. Each episode includes:

- A session ID
- A task description and the task category it belongs to. The possible task categories are: Pick and Place, Open / Close, Move / Slide, Knock Over / Topple, Cover / Drape / Fold, Group / Organize / Stack, Find / Search, Minimal or No Action, Object Manipulation, Sorting / Classification, Tool Use.
- A scene and task analysis
- Head-to-head evaluation results

Using the episode data provided, generate a **structured and comprehensive summary report** in the format below:

1. **Policy Overview**
A brief paragraph summarizing the general behavior, capabilities, and limitations of the policy.

2. **Comparative Performance**
How the policy performed in head-to-head comparisons against other policies across the different task categories. For each task category, create a bullet point with a discussion of how the policy consistently outperformed or underperformed compared to all the other policies. Make sure in this section that every claim about the policy is with respect to other competing policies. When making a claim, always mention how the other policies performed in comparison to the current policy. Do not discuss the policy in isolation. Don't mention a task category unless there is evidence of the policy performing well or poorly in that category across multiple episodes. Make your claims based on overall performance or underperformance for specific task categories rather than individual episodes. There is no need to reference specific session IDs in this section (no `<ref>` tags).

3. **Strengths**
Bullet-pointed list of notable strengths in manipulation behavior or general reliability. Mention the task categories the policy is good at (if any) instead of basing a claim on a single instance. Focus on generalizable behaviors like smooth trajectories, robust grasping, or adaptability. Use concrete examples and session ID citations.

4. **Weaknesses**
Bullet-pointed list of recurring limitations or error patterns. Mention the task categories the policy is poor at instead of basing a claim on a single instance. Mention issues such as fine motor control, object confusion, multi-step failure, etc. Include session ID references with `<ref>` tags.

5. **Instruction Following**
Analyze how well the policy understands and executes task instructions. Note sensitivity to language structure, ability to follow negated or relational commands, issues with ambiguous phrasing, ability to handle typos, etc. Cite session-specific evidence.

6. **Reasoning**
Evaluate the policy's ability to reason about both the **scene context** (e.g., spatial relationships, object visibility) and the **text instruction** (e.g., goal inference, conditional logic). Mention cases where reasoning appears strong or deficient. Use `<ref>` tags to support your analysis.

7. **Manipulation Skills**
Describe the physical performance of the policy: grasping, placing, stacking, inserting, pouring, drawer use, and recovery from errors. Use examples to show when skills succeed or fail.

8. **Robustness to Scene Variations**
Assess the policy's performance under different lighting, clutter levels, object positions, and camera views. Note any sensitivities to occlusion or distractors, etc.

9. **Common Failure Modes**
List frequently observed failures (e.g., freezing mid-task, grabbing wrong item, failing passive commands). Provide short descriptions and supporting citations.

Instructions:

- When referring to a session, always cite the full session ID (UUID format, e.g., `16e5bbda-57c1-4e58-a24a-b39ee8142d41`) exactly as provided. Do not shorten, truncate or modify it in any way.
- Always wrap session IDs inside `<ref>...</ref>` tags. Example: `<ref>16e5bbda-57c1-4e58-a24a-b39ee8142d41</ref>`
- Try to cite as many session IDs as possible to support your claims, but only if they are relevant to the point you're making.
- Avoid generalizing from a single episode unless there is clear evidence of a pattern.
- Keep the tone analytical and professional, emphasizing repeatable behaviors and insights.
- Do not invent session IDs. Only use session IDs present in the provided episode reports.
- There is no need to mention the specific number of episodes and wins/losses/ties in head-to-head evaluations in this report.

The individual episode reports are as follows:

```
========== Episode Report #1 ==========
```
...

Figure 14: The prompt template used to generate the full policy performance reports.

Given the following full evaluation report of a robot manipulation policy, generate a concise, high-quality summary that captures the main findings from sections 1 through 9.

Each bullet should summarize the corresponding section in a few sentence fragments, focusing on the most important points. Avoid excessive detail, ensure clarity and correctness. Stick to the facts presented in the full report.

Use the following format exactly:

- Policy Overview: <summary>

- Comparative Performance: <summary>

- Strengths: <summary>

- Weaknesses: <summary>

- Instruction Following: <summary>

- Reasoning: <summary>

- Manipulation Skills: <summary>

- Robustness to Scene Variations: <summary>

- Common Failure Modes: <summary>

Place a line break between each bullet point. Don't output anything before or after the bullet points.

Here is the full report to summarize:

`FULL REPORT`

Figure 15: The prompt template used to generate a concise summary of the full policy performance report.