

What Are We Actually Benchmarking in Robot Manipulation?

Tianchong Jiang¹ Tewodros W. Ayalew^{2,*} Luzhe Sun^{1,*}
Xiangshan Tan^{1,*} Samuel Wheeler^{3,*} Matthew Walter¹

¹Toyota Technological Institute at Chicago, Chicago, IL, USA

²University of Chicago, Chicago, IL, USA

³Argonne National Laboratory, Lemont, IL, USA

{tianchongj, luzhesun, vincenttann, mwalter}@ttic.edu

tewodrosayalew@uchicago.edu, swwheeler@anl.gov

*Equal contribution; listed alphabetically.

Abstract: A robotics benchmark score measures success under one fixed evaluation setup, yet is routinely treated as evidence of general manipulation capability. We identify four failure modes, each of which weakens or invalidates a benchmark’s role as a valid proxy for that capability: shortcut solvability, lack of statistical significance, creeping overfitting, and data-source dependence. We propose one diagnostic per failure mode. We audit LIBERO, CALVIN, SimplerEnv, RoboCasa, and RoboTwin 2.0 under these diagnostics. LIBERO and CALVIN fail multiple diagnostics. RoboCasa and RoboTwin 2.0 fail fewer, despite appearing far less often in recent progress claims. On LIBERO, a 0.09B probe with no language encoder scores at or near reported SOTA, and most reported gains are not provably statistically significant. On CALVIN, randomizing block poses within the training range drops performance for every tested policy. We release the four diagnostics with reference implementations for authors and reviewers to apply before treating a benchmark score as evidence of progress.

Keywords: Manipulation benchmarks, Benchmark validity, Policy evaluation

1 Introduction

In March 2026 alone, 79 arXiv papers reported results on LIBERO (Figure 2). When a benchmark is used at this rate, it is not only a testbed for individual researchers but also the field’s definition of progress [1].

A benchmark score is not an end in itself. It is a proxy for general capabilities that we want robots to have in the real world, like tidying our homes, cleaning and folding laundry, and assisting the elderly. A benchmark is meaningful when a high score is evidence of that capability. It is therefore important to test whether the most-reported manipulation benchmarks are meaningful.

The ideal test would train policies on shared demonstrations, evaluate them on a set of representative real-world tasks, and check whether the relative ranking of policies on the benchmark matches that on real-world tasks. Such a test is impractical: there is no agreed-upon real-world task set to evaluate on, and retraining each policy on shared demonstrations is too expensive. Another approach is to check whether policies that rank highly on one benchmark also rank highly on others [1]. This does not work either: Authors choose which benchmarks to report and tune their methods to reach SOTA on those, so new papers almost always claim SOTA on every benchmark they report.

We instead define four diagnostics that provide insight into the extent to which performance on a benchmark is evidence of general manipulation capability. We apply them to the five most widely reported manipulation benchmarks, LIBERO [2], CALVIN [3], SimplerEnv [4], RoboCasa [5], and RoboTwin 2.0 [6].

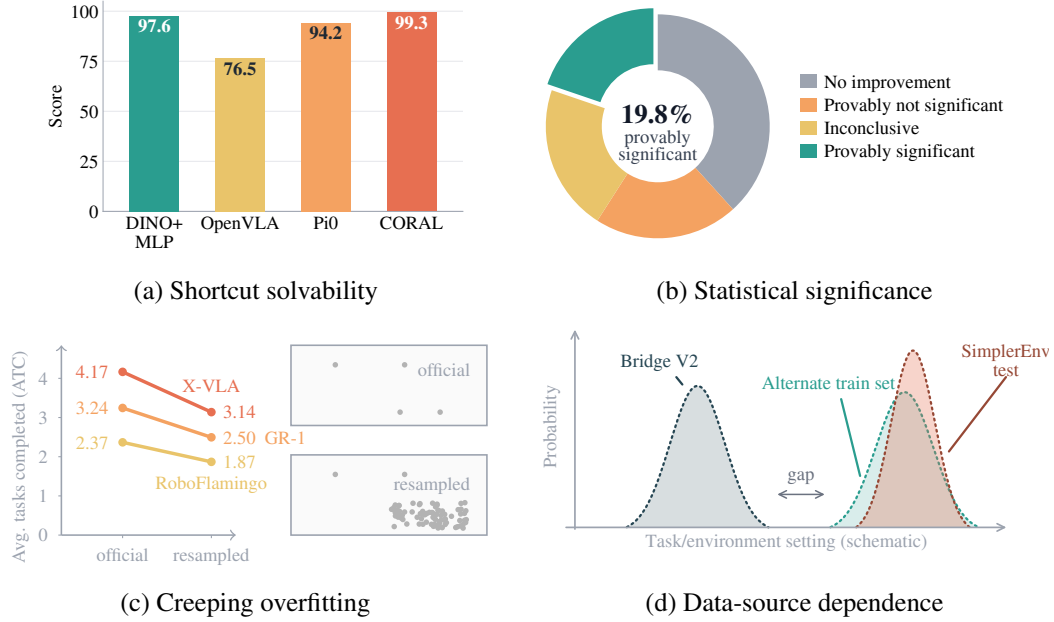


Figure 1: The panels preview the four diagnostics used in our audit. (a) A 0.09B-parameter probe nearly matches SOTA on LIBERO average. (b) Only 19.8% of LIBERO SOTA claims are provably significant. (c) On CALVIN, resampling block poses within the training distribution lowers average tasks completed (ATC). (d) On SimplerEnv, the score cannot distinguish a policy that generalizes from one trained on data close to the test.

Shortcut solvability. A benchmark score is evidence of capability only if any policy achieving it has the capability. We show that for LIBERO, a simple 0.09B probe with no language encoder and no large-scale robotics pretraining scores at or near the best reported result, while on RoboTwin 2.0 and RoboCasa it scores well below the best reported result.

Statistical significance. Most benchmarks report only an aggregate success rate, from which the significance of a reported gain cannot be tested. On LIBERO and SimplerEnv, only 19.8% and 19.7% of SOTA claims, respectively, are provably significant. On RoboCasa and RoboTwin 2.0, the shares rise to 53.3% and 73.7%, respectively.

Creeping overfitting. A benchmark’s test distribution can be a narrow region of the training distribution’s support, and its fixed test set is one sample from that region. Policies can fit either, which we call *distribution overfitting* and *sample overfitting*. On CALVIN, resampling block poses within the training range lowers average tasks completed out of five (ATC) from 4.17 to 3.14.

Data-source dependence. A small model trained on demonstrations closer to the test distribution can match a much larger model trained on more general data. As a result, a benchmark score cannot distinguish generalization from how close the training data is to the test distribution.

Across these four diagnostics, the most-reported benchmarks fail the most. LIBERO, CALVIN, and SimplerEnv each fail several diagnostics, while RoboTwin 2.0 and RoboCasa fare better.

We release reference implementations of the four diagnostics. Authors of new benchmarks can run them before publishing. For the five benchmarks we audit, our per-benchmark results help authors and reviewers interpret reported gains. Code and rollout results are in the supplementary material.

2 Related Work

2.1 Benchmark Audits in Computer Vision and NLP

There is a rich literature auditing benchmark validity in computer vision and natural language processing, and each of the four failure modes we apply to manipulation has direct precedent there.

Shortcut solvability. Goyal et al. [7] showed that the original visual question answering (VQA) benchmark allowed models to answer questions starting “Do you see a . . .” with 87% accuracy by blindly answering “yes”, without reading the rest of the question or looking at the image. They built the VQA v2 benchmark with complementary-image pairs to remove this language-prior shortcut. Lapuschkin et al. [8] found that models trained on PASCAL VOC could predict “horse” from a copyright tag in the corner of the image. Analogously, on LIBERO and CALVIN, a probe with no language encoder can do relatively well.

Statistical significance. Everingham et al. [9] applied the Friedman/Nemenyi test to PASCAL VOC and found no statistically significant difference among the top six competing methods. Musgrave et al. [10] later showed that a subfield’s reported gains did not survive fair-comparison auditing.

Creeping overfitting. Creeping overfitting is when models become too adapted to a benchmark over time, memorizing its idiosyncrasies and failing to generalize [1, 11]. Torralba and Efros [1] showed this across popular vision datasets, where classifiers trained on one dataset transferred poorly to others, with average drops in accuracy near 50%. Recht et al. [12] asked whether models had fit the one fixed test set. They rebuilt a fresh test set under the original ImageNet protocol and found rankings largely unchanged, so little sign of fitting to that set. Hendrycks and Dietterich [13] asked whether models had instead fit the narrow test distribution. They corrupted the inputs and found accuracy dropped sharply, a clear sign of fitting to that distribution. We run both checks on manipulation benchmarks.

Data-source dependence. Several audits show that test-set scores partly reflect train-test proximity rather than learned capability. Barz and Denzler [14] found that CIFAR-10 and CIFAR-100 test sets contain near-duplicates of training images, so reported accuracy partly measures memorization rather than generalization. Elangovan et al. [15] document analogous train/test leakage across NLP benchmarks. Lewis et al. [16] show that most test questions in popular open-domain QA datasets have an answer or near-paraphrase in the training set, and models score much worse on the leakage-free subset. We extend this to robot manipulation. A small model trained on data close to the test distribution can outperform a larger model trained on more distant data.

2.2 Manipulation Benchmarks and Prior Audits

LIBERO [2] is a language-conditioned manipulation benchmark with four task suites. CALVIN [3] is a long-horizon language-conditioned manipulation benchmark with held-out instructions. SimplerEnv [4] reproduces real Bridge and Google-robot setups in simulation. RoboCasa [5] is a kitchen-task benchmark with procedurally generated scenes and objects. RoboTwin 2.0 [6] is a dual-arm manipulation benchmark with strong domain randomization for bimanual manipulation.

Several prior works audit specific manipulation benchmarks. LIBERO-Pro [17] and LIBERO-Plus [18] change LIBERO scenes and instructions and show that high scores drop. But their changed inputs fall outside the benchmark’s training distribution, so a drop might be the result of weak generalization. Our diagnostic instead keeps every input inside the training distribution, so a drop measures creeping overfitting rather than weak generalization.

Many other manipulation simulators [19–21] and benchmarks [22–33] exist. We focus on the five in Figure 2 since they appear far more often in recent VLA and generalist-policy work (Appendix D).

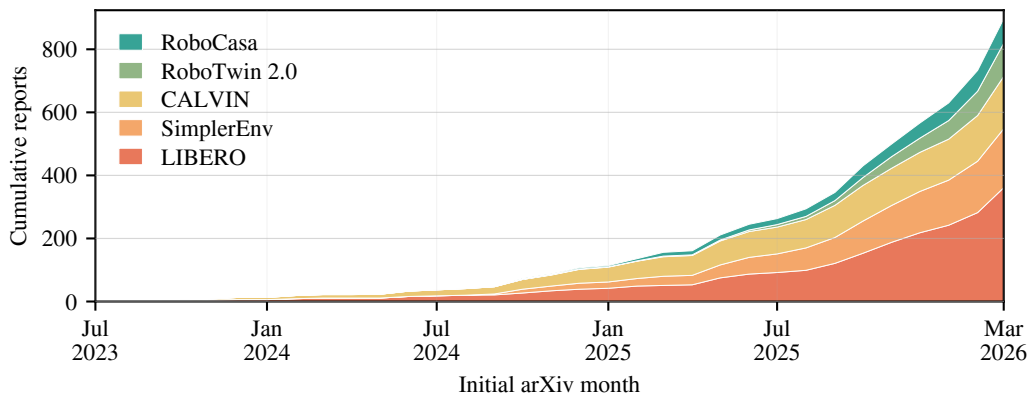


Figure 2: Cumulative counts of arXiv papers reporting results on CALVIN, LIBERO, SimplerEnv, RoboCasa, and RoboTwin 2.0. Sources, deduplication keys, and per-paper classification criteria are in Appendix D.

3 Shortcut Solvability

A benchmark score is evidence of capability only if every policy that reaches a comparable score has the capability. We call a policy that reaches the score while lacking the capability a *shortcut*, and a benchmark *shortcut-solvable* when one exists.

LIBERO [2] is a language-conditioned manipulation benchmark that tests lifelong learning across language-specified tasks. Scoring well is meant to require understanding the instruction, yet a high score is read as evidence of much more. Because the top of its leaderboard is held by foundation VLAs, a high score is taken to show that the policy generalizes broadly, and that reaching it requires large-scale robotics pretraining and a large, expressive model.

We test whether a high score can be reached without those properties. Our probe is a deliberately small model with approximately 90M parameters (a shared DINOv2 encoder [34] and a small MLP head), with no language encoder, no large-scale robotics pretraining, and no expressive action head (i.e., no diffusion head or action tokenizer). Because each benchmark draws its instructions from a fixed public set, the probe replaces the instruction with a learned per-task embedding that it looks up by index instead of reading as text. On LIBERO each instruction maps to its own embedding, and on CALVIN several phrasings of a subtask map to the same embedding. The probe uses only what each benchmark’s evaluation protocol lets any policy use, so reaching its score this way reveals a weakness in the protocol, not a trick of ours.

Table 1 shows that LIBERO is shortcut-solvable since the probe performs within one percentage point of the best published result on three of four splits.¹ On CALVIN, the shortcut is real but smaller, since the probe matches the performance of widely-used policies but not the current best. On RoboTwin 2.0, RoboCasa, and SimplerEnv, the probe scores well below the best, indicating the lack of a shortcut.

We claim only that a high score on LIBERO is not, on its own, evidence of general manipulation capability—our probe achieves a high score yet lacks such a capability. We do not claim that a high-scoring policy lacks that capability, since a capable policy reaches a high score too.²

¹We train and tune the probe per suite and report the best of several checkpoints evaluated on the suite. See Appendix A.1.

²Suppose that one can pass a written driving test by memorizing the answer key. Passing this test is not evidence of driving ability. However, the people who pass the test can still be good drivers, just like the models that do well on LIBERO can still be capable.

Table 1: Shortcut-solvability probe results across five benchmarks.

Benchmark	Protocol/Suite	DINO+MLP	Foundation VLA	Reported (Δ_F)	Best Reported	Reported (Δ_B)
LIBERO	Spatial	99.0%	OpenVLA (7B)	84.7% (-14.3)	π_0 +T-MEE (2.3B)	99.8% ($+0.8$)
	Object	100.0%	OpenVLA (7B)	88.4% (-11.6)	$\pi_{0.5}$ +MoH (3B)	100.0% ($+0.0$)
	Goal	98.8%	OpenVLA (7B)	79.2% (-19.6)	MoLA (1.64B)	99.5% ($+0.7$)
	Long	92.4%	OpenVLA (7B)	53.7% (-38.7)	CORAL _{simVLA} (0.8B)	98.8% ($+6.4$)
CALVIN [†]	D→D	3.123	DeeR-VLA (3B)	2.83 (-0.29)	MDT-V (n.d.)	4.52 ($+1.40$)
	ABC→D	3.242	RoboFlamingo (3B)	2.48 (-0.76)	MMaDA-VLA (8B)	4.78 ($+1.54$)
	ABCD→D	3.872	RoboFlamingo (3B)	4.09 ($+0.22$)	Xiaomi-0 (4.7B)	4.80 ($+0.93$)
SimplerEnv	WidowX	0.0%	X-VLA (0.9B)	95.8% ($+95.8$)	CORAL _{simVLA} (0.8B)	97.9% ($+97.9$)
RoboTwin 2.0	Clean (50+500)	60.4%	X-VLA (0.9B)	72.8% ($+12.4$)	MotuBrain (n.d.)	95.8% ($+35.4$)
	Randomized (50+500)	59.4%	X-VLA (0.9B)	72.8% ($+13.4$)	MotuBrain (n.d.)	96.1% ($+36.7$)
RoboCasa	RSS24	18.8%	GR00T-N1 (2B)	49.6% ($+30.8$)	X-WAM (n.d.)	79.2% ($+60.4$)

Foundation VLA: most-cited protocol-comparable foundation VLA reporting on the protocol (chosen per protocol for CALVIN).

Best reported: top non-RL-finetuned result (RL-finetuned excluded, as the probe is imitation-only).

Δ_F and Δ_B : the Foundation-VLA score and the Best-reported score, respectively, minus the DINO+MLP probe score (ATC for CALVIN, percentage points otherwise); a negative value means the probe scores higher.

CALVIN[†] reports ATC (out of 5). “n.d.” = params not disclosed. Policy sources are listed in Appendix A.1.

4 Statistical Significance

When a new policy is published, it will often report a higher aggregate benchmark score than the previous best from the literature, and that gap is taken as progress. In this section, we consider the question of determining how statistically significant such performance gaps may be.

Most papers report only an aggregate score, most commonly an average success rate. However, the aggregate score alone is often not enough to determine the statistical significance of a reported performance gap, a gap we then call *inconclusive*. It arises because the variance of the empirically observed performance gap is a function of task-level and sample-level information that is not retained by the average performance statistic (Appendix C).

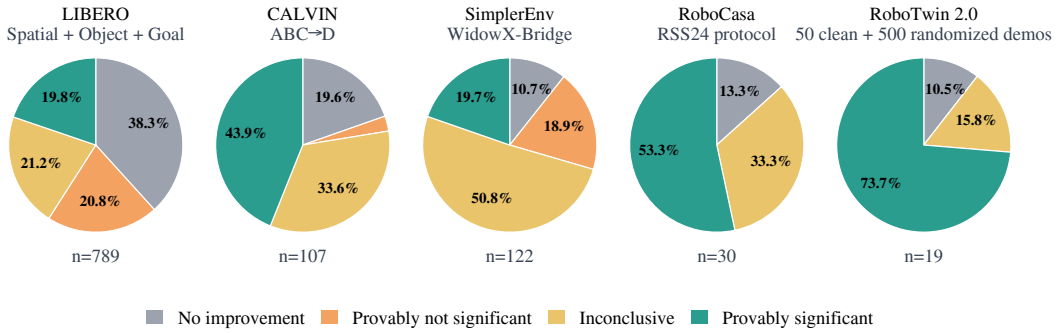


Figure 3: Reported gains from public scores alone. Each pie sorts one benchmark’s previous-best-to-new comparisons into the legend’s four groups (bounds derived in Appendix C). Each comparison uses the previous best as reported in the new policy’s own paper, rather than the highest earlier score we could find on arXiv. LIBERO pools three suites. n is the number of comparisons. All bounds use a 5% significance level.

Method. We compare a new policy B against a previously published policy A with a lower aggregate score. Each benchmark consists of T tasks, with S paired samples per task. After evaluating both policies on the benchmark, a set of observed outcomes indexed by task and sample is collected, with $Y_{A,t,i}, Y_{B,t,i} \in \{0, \dots, R\}$ for task $t \in 1, \dots, T$ and sample $i \in 1, \dots, S$. We assume that each policy is published with an average score $\hat{\mu}_A, \hat{\mu}_B$, where $\hat{\mu}_A < \hat{\mu}_B$.

Define the empirical gap $\hat{\Delta} := \hat{\mu}_B - \hat{\mu}_A$. When both policies are evaluated on the same benchmark instances, we use a one-sided paired Wald test with a task-stratified variance estimator. Let

$$\delta_{t,i} = Y_{B,t,i} - Y_{A,t,i}, \quad d_t = \sum_{i=1}^S \delta_{t,i}, \quad s_t = \sum_{i=1}^S \delta_{t,i}^2.$$

The paired stratified Wald statistic for testing $H_0 : \Delta \leq 0$ against $H_1 : \Delta > 0$, where Δ denotes the benchmark-average improvement in the top-line metric, is

$$Z = \frac{TS(\hat{\mu}_B - \hat{\mu}_A)}{\sqrt{\frac{S}{S-1} \sum_{t=1}^T (s_t - d_t^2/S)}}.$$

The paired Wald statistic cannot be computed from top-line metrics alone. Lacking access to the full paired outcome data, we derive two cutoffs on the reported empirical gap. The rejection-feasibility threshold $\delta_{\exists}(\hat{\mu}_A; T, S, R, \alpha)$ is chosen so that if $\hat{\Delta} < \delta_{\exists}(\hat{\mu}_A; T, S, R, \alpha)$, then no paired outcome table compatible with the reported top-line metrics can reject the null hypothesis. The rejection-guarantee threshold $\delta_{\forall}(\hat{\mu}_A; T, S, R, \alpha)$ is chosen such that $\hat{\Delta} \geq \delta_{\forall}(\hat{\mu}_A; T, S, R, \alpha)$, implies that every paired outcome table compatible with the reported top-line metrics rejects the null hypothesis. The derivation is given in Appendix C.

For paired evaluations, these top-line bounds are sharp. Below the feasibility threshold, rejection is impossible from the reported aggregate scores. Above the guarantee threshold, rejection is forced by the reported aggregate scores. Between the corresponding pointwise feasibility and guarantee conditions, the aggregate scores alone do not determine significance: the same reported counts are compatible with paired outcome tables that reject and with paired outcome tables that do not reject.

Some audited papers do not evaluate both policies on the same benchmark instances, choosing instead to evaluate on iid samples drawn from the benchmark distribution. For these comparisons, the natural reference procedure is a stratified two-sample Wald test rather than the paired Wald test above. We still use the paired envelopes as conservative top-line cutoffs, but in such cases the thresholds will no longer be sharp.

Figure 3 shows the provably significant share is small on LIBERO and SimplerEnv and much larger on RoboCasa and RoboTwin 2.0. This is partly because RoboCasa and RoboTwin 2.0 have fewer reported results, so the scores sit further apart and larger gaps are easier to prove significant.

5 Creeping Overfitting

A benchmark has a fixed and usually narrow test distribution (for example, over object types, object poses, backgrounds, and camera poses). Its official test set is a set of finite samples drawn from that distribution. Over time, models can become too tuned to the benchmark rather than acquiring broader manipulation skill, which prior work calls *creeping overfitting* [11]. Fitting to the narrow test distribution is *distribution overfitting*, and fitting to the fixed samples is *sample overfitting*.

Distribution overfitting. CALVIN’s most commonly used protocol, ABC→D, trains on three table scenes (A, B, C) and tests on a fourth held-out scene D, where each evaluation episode is a chain of five language-conditioned tasks scored as the average number completed (ATC). During training the table blocks can start at any pose in a range, but the released scene-D evaluation fixes them at the same poses. We resampled the block poses from that same training range and held everything else fixed. ATC fell by 1.03 for X-VLA [35], 0.75 for GR-1 [36], and 0.50 for RoboFlamingo [37] with 5/5-chain drops of 25.0%, 13.5%, and 6.4% (Figure 1c). Later policies show greater drops.

On SimplerEnv, we changed the WidowX stack task in three ways (Figure 4). We measured CogACT-Base [38], InternVLA-M1 [39], X-VLA-WidowX [35], and Dexbotic / DB-MemVLA [40] against a matched local calibration, our rerun on the official WidowX stack protocol (Appendix A.2). We reversed the instruction (yellow on green instead of green on yellow), stacked the target block on a support block instead of the table, and resampled block positions and the initial arm pose. All

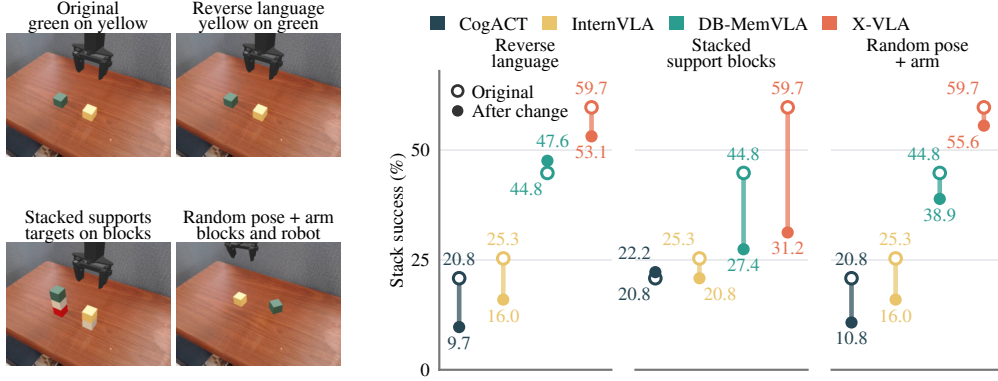


Figure 4: SimplerEnv stack task with three changes that stay inside the training distribution. Left, rendered initial states for the original protocol and the three changes. Right, original fixed-grid stack success and success after each change.

three changes already appear in the training set about as often as the official conditions do (Appendix A.2). So they do not make the test more out-of-distribution than those conditions, and any drop shows overfitting rather than weak generalization.

The lower-scoring policies (CogACT-Base, InternVLA-M1) drop most on the reversed instruction, while the higher-scoring policies (X-VLA-WidowX, Dexbotic) hold there but drop sharply under the stacked-support change. The benchmark uses a single fixed camera, which makes a block’s depth and height hard to recover and may force policies to assume every block sits on the table. The stacking drop is consistent with that assumption.

Sample overfitting. We redrew the fixed sample on LIBERO and CALVIN. On LIBERO, redrawing the init-state file from the same generator with 10,000 rollouts per policy moved success by less than a percentage point against a matched local calibration (Spatial Forcing [41] -0.62 , SimVLA [42] $+0.30$, $\pi_{0.5}$ LeRobot [43, 44] -0.14). On CALVIN, drawing two fresh sequence manifests from the official generator moved ATC by at most 0.11 for the CALVIN policies above (X-VLA, GR-1, and RoboFlamingo), an order of magnitude below the drops we saw when we resampled the poses. Closed-loop rollouts are not bitwise deterministic (Appendix B), so each calibration carries noise that leaves the sign of any one policy’s move uncertain. Either way the moves are small, no larger than that noise and far below the pose-resampling drops on the same CALVIN policies.

6 Data-Source Dependence

A rising success rate is read across the field as evidence of a more capable policy. This section asks when that reading can be inflated by the choice of training data rather than earned by capability. The risk exists only when the training data sits far from the test, because then a high score looks like generalization across the gap. For LIBERO, CALVIN, RoboCasa, and RoboTwin 2.0 the training data covers the test or the gap is small, so this diagnostic does not apply.

SimplerEnv is the exception. Its test runs in a simulated WidowX setup, but the training data is real-world BridgeData V2 demonstrations [45] far from that test. The gap is real, so a high score reads as generalization across it.

However, SimplerEnv does not restrict where training data comes from. Without that restriction, the gap can be removed instead of crossed, by collecting training data right next to the test. We show this with an extreme case. X-VLA, a 0.9B model with broad real and simulated pretraining, reports 95.8% (92/96) [35], the field’s reference for a near-solved score. We train a separate 22M policy per task, with no robotics pretraining, on 120 scripted expert demonstrations recorded in simulation right beside the official test. Together they reach 94.8% (91/96).

We do not claim the small policy is better, only that on a benchmark with unrestricted training data a high score alone is not evidence of capability. A genuinely strong method could reach the same score through real generalization, and the score alone cannot tell the two apart. The fix needs to come from the benchmark itself, either by restricting the training set or by asking each submission to report how close its training data sits to the test.

7 Discussion

All four failure modes share one root cause: a benchmark score is an empirical average of success over a fixed test distribution, treated as evidence of capability over the much broader distribution of interest. Each diagnostic catches one way the score can detach from the capability it stands in for: the test instances admit a shortcut, the empirical average is too noisy to resolve the claimed gap, the test distribution is a narrow slice of the variation we care about, or the gap between training and test data drives the score more than capability does.

What a future manipulation benchmark should look like. Our diagnostics point to two design goals for future manipulation benchmarks.

First, the test distribution should cover broad variation in objects and scenes. Our creeping-overfitting results on CALVIN and SimplerEnv, together with prior LIBERO-Pro and LIBERO-Plus findings on LIBERO [17, 18], show that a high score on a narrow test distribution is not evidence that a policy performs at the same level under broad variation in objects and scenes. A wider distribution leaves less room for a brittle policy to score well without generalizing. The shortcut diagnostic results align with this. The 0.09B DINO+MLP baseline matches SOTA on LIBERO and CALVIN but trails by more than 30% on RoboTwin 2.0, whose object and scene variation is much greater. Difficulty should come from this variation, not from how far the training data is from test. Section 6 shows that a 22M policy matches a 0.9B foundation VLA simply by training on demonstrations collected next to the test, so a benchmark whose difficulty rests on that distance can be gamed.

Second, the official protocol should state what score gap counts as a statistically significant improvement over the previous SOTA, so that gains smaller than evaluation noise are not reported as progress. The protocol should also use a test set large enough to detect the gains authors want to claim. When the test set is fixed, the protocol should require each policy’s per-instance outcomes, not only the aggregate score. With those outcomes anyone can run the paired test, which can prove significance at far smaller gaps than the aggregate score allows.

Until such benchmarks exist. The benchmarks we audit remain perfectly useful for everyday development work, such as fast iteration, debugging, and comparison against one’s own earlier runs. For shortcut solvability and data-source dependence, a high score is not by itself evidence of the capability the benchmark is taken to measure. Creeping overfitting and weak significance weaken a reported result rather than invalidate it. Either way, the result alone does not establish progress for the field. Authors who think a commonly-reported benchmark fails one of our diagnostics can use this paper to support the argument that they should not be required to report on that benchmark, and that a benchmark they consider stronger is an acceptable substitute. Reviewers can use it in the opposite direction, to question a small gain claimed as progress on such a benchmark.

Beyond fixed-population averages. The diagnostics in this paper audit benchmarks that score a policy by its empirical average over a fixed test distribution. A line of work asks whether that scoring rule can be replaced. Approaches based on item response theory model success as a function of a policy’s effectiveness relative to the difficulty of a given instance [46, 47]. Combined with adaptive selection of test instances by this difficulty [48, 49], this direction could address creeping overfitting, data-source dependence, and statistical-power constraints jointly rather than one at a time.

8 Limitations

The ideal test would train many policies on one shared set of demonstrations, evaluate them on a large test set that samples the capabilities we care about, and check whether the benchmark’s policy ranking matches the ranking on that sample. The Introduction explains why that test is impractical. Our four diagnostics are a practical substitute.

For some diagnostics, like creeping overfitting, we could run only a few policies. Reproducing a policy needs both its open-source weights and the evaluation script that produced its reported numbers. Most policies do not release both, so we cannot reproduce them.

9 Conclusion

A benchmark score is useful only if it provides evidence of the capability we care about. We identify four ways in which a high score can fail to be that evidence, and provide a diagnostic for each. We ran the diagnostics on five widely used manipulation benchmarks. The most-reported ones failed the most diagnostics. We release the diagnostics so authors and reviewers can run them before reporting a score. Code, artifacts, and per-rollout outcomes are in the supplementary material.

Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

- [1] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528, 2011.
- [2] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [3] 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*, 7(3):7327–7334, 2022.
- [4] X. Li, K. Hsu, J. Gu, O. Mees, K. Pertsch, 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. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 3705–3728, 2025.
- [5] 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. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.
- [6] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Z. Li, Q. Liang, X. Lin, Y. Ge, Z. Gu, W. Deng, Y. Guo, T. Nian, X. Xie, Q. Chen, K. Su, T. Xu, G. Liu, M. Hu, H.-a. Gao, K. Wang, Z. Liang, Y. Qin, X. Yang, P. Luo, and Y. Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.
- [7] Y. Goyal, T. Khot, A. Agrawal, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. *International Journal of Computer Vision*, 127(4):398–414, 2019.

- [8] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1096, 2019.
- [9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.
- [10] K. Musgrave, S. Belongie, and S.-N. Lim. A metric learning reality check. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 12370 of *Lecture Notes in Computer Science*, pages 681–699. Springer, 2020. doi:10.1007/978-3-030-58595-2_41.
- [11] J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszałek, C. Schmid, B. C. Russell, A. Torralba, C. K. I. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. In *Toward Category-Level Object Recognition*, Lecture Notes in Computer Science, pages 29–48. Springer, 2006.
- [12] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do ImageNet classifiers generalize to ImageNet? In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97, pages 5389–5400, 2019.
- [13] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [14] B. Barz and J. Denzler. Do we train on test data? Purging CIFAR of near-duplicates. *Journal of Imaging*, 6(6):41, 2020.
- [15] A. Elangovan, J. He, and K. Verspoor. Memorization vs. generalization: Quantifying data leakage in NLP performance evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1325–1335, 2021.
- [16] P. Lewis, P. Stenetorp, and S. Riedel. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1000–1008, 2021.
- [17] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.
- [18] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, J. Li, X. He, S. Zhang, Z. Fei, J. Fu, J. Gong, and X. Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.
- [19] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.
- [20] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su. ManiSkill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Datasets and Benchmarks Track.
- [21] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D’Arpino, S. Buch, S. Srivastava, L. P. Tchapti, M. E. Tchapti, K. Vainio, J. Wong, L. Fei-Fei, and S. Savarese. iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

- [22] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1094–1100, 2020.
- [23] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 155 of *Proceedings of Machine Learning Research*, pages 726–747. PMLR, 2021.
- [24] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. VIMA: General robot manipulation with multimodal prompts. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 14975–15022, 2023.
- [25] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, pages 1–8, 2024.
- [26] R. Gong, J. Huang, Y. Zhao, H. Geng, X. Gao, Q. Wu, W. Ai, Z. Zhou, D. Terzopoulos, S.-C. Zhu, B. Jia, and S. Huang. ARNOLD: A benchmark for language-grounded task learning with continuous states in realistic 3D scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20483–20495, 2023.
- [27] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao, Z. Fei, Z. Yin, Z. Wu, Y.-G. Jiang, and X. Qiu. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11142–11152, 2025.
- [28] 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.
- [29] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. THE COLOSSEUM: A benchmark for evaluating generalization for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.
- [30] M. Heo, Y. Lee, D. Lee, and J. J. Lim. FurnitureBench: Reproducible real-world benchmark for long-horizon complex manipulation. *International Journal of Robotics Research*, 44(10–11):1863–1891, 2025.
- [31] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [32] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei. BEHAVIOR-1K: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 80–93, 2023.
- [33] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [34] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut,

- A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.
- [35] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng, Y. Zheng, J. Zou, Y. Chen, J. Zeng, Y.-Q. Zhang, J. Pang, J. Liu, T. Wang, and X. Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.
- [36] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- [37] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong. Vision-language foundation models as effective robot imitators. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- [38] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.
- [39] X. Chen, Y. Chen, Y. Fu, N. Gao, J. Jia, W. Jin, H. Li, Y. Mu, J. Pang, Y. Qiao, Y. Tian, B. Wang, B. Wang, F. Wang, H. Wang, T. Wang, Z. Wang, X. Wei, C. Wu, S. Yang, J. Ye, J. Yu, J. Zeng, J. Zhang, J. Zhang, S. Zhang, F. Zheng, B. Zhou, and Y. Zhu. InternVLA-M1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.
- [40] B. Xie, E. Zhou, F. Jia, H. Shi, H. Fan, H. Zhang, H. Li, J. Sun, J. Bin, J. Huang, K. Liu, K. Liu, K. Gu, L. Sun, M. Zhang, P. Han, R. Hao, R. Zhang, S. Huang, S. Xie, T. Wang, T. Liu, W. Tang, W. Zhu, Y. Chen, Y. Liu, Y. Zhou, Y. Liu, Y. Zhao, Y. Ma, Y. Wei, Y. Chen, Z. Chen, Z. Li, Z. Wu, Z. Zhang, Z. Liu, Z. Yan, and Z. Zhang. Dexbotic: Open-source vision-language-action toolbox. *arXiv preprint arXiv:2510.23511*, 2025.
- [41] F. Li, W. Song, H. Zhao, J. Wang, P. Ding, D. Wang, L. Zeng, and H. Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.
- [42] Y. Luo, W. Chen, T. Liang, B. Wang, and Z. Li. SimVLA: A simple VLA baseline for robotic manipulation. *arXiv preprint arXiv:2602.18224*, 2026.
- [43] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.
- [44] R. Cadene, S. Alibert, F. Capuano, M. Aractingi, A. Zouitine, P. Kooijmans, J. Choghari, M. Russi, C. Pascal, S. Palma, D. Aubakirova, M. Shukor, J. Moss, A. Soare, Q. Lhoest, Q. Gallouédec, and T. Wolf. LeRobot: An open-source library for end-to-end robot learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.
- [45] 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, A. Lee, K. Fang, C. Finn, and S. Levine. BridgeData V2: A dataset for robot learning at scale. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1723–1736, 2023.

- [46] F. Martínez-Plumed, R. B. C. Prudêncio, A. Martínez-Usó, and J. Hernández-Orallo. Item response theory in AI: Analysing machine learning classifiers at the instance level. *Artificial Intelligence*, 271:18–42, 2019.
- [47] P. Rodriguez, J. Barrow, A. Hoyle, J. P. Lalor, R. Jia, and J. Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4486–4503, 2021.
- [48] D. J. Weiss and G. G. Kingsbury. Application of computerized adaptive testing to educational problems. *Journal of Educational Measurement*, 21(4):361–375, 1984.
- [49] H. Song and P. Flach. Efficient and robust model benchmarks with item response theory and adaptive testing. *International Journal of Interactive Multimedia and Artificial Intelligence*, 6(5):110–118, 2021.
- [50] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 2679–2713. PMLR, 2025.
- [51] Y. Li, B. Zhang, C. Gu, Z. Ma, J. Zhang, J. Deng, X. Zhu, and L. Zhang. From imagined futures to executable actions: Mixture of latent actions for robot manipulation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.
- [52] Y. Luo, W. Chen, T. Liang, and Z. Li. CORAL: Scalable multi-task robot learning via LoRA experts. *arXiv preprint arXiv:2603.09298*, 2026.
- [53] Y. Yang, S. Zeng, T. Lin, X. Chang, D. Qi, J. Xiao, H. Liu, R. Chen, Y. Chen, D. Huo, F. Xiong, X. Wei, Z. Ma, and M. Xu. ABot-M0: VLA foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.
- [54] MotuBrain Team, C. Xiang, F. Bao, H. Liu, H. Tan, H. Bi, J. Li, J. Liu, J. Pang, K. Jing, L. Liu, M. Cai, R. Cui, R. Zhao, R. Wang, S. Huang, Y. Feng, Y. Rong, Z. Wang, and J. Zhu. Mo-tuBrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026.
- [55] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- [56] NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. 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.
- [57] S. Bai, D. Wang, C. Chi, W. Zhou, J. Lyu, X. Zhao, P. Wang, Z. Wang, L. Xing, S. Zhang, and B. Chen. Reshaping action error distributions for reliable vision-language-action models. *arXiv preprint arXiv:2602.04228*, 2026.
- [58] D. Jing, G. Wang, J. Liu, W. Tang, Z. Sun, Y. Yao, Z. Wei, Y. Liu, Z. Lu, and M. Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.
- [59] M. Reuss, O. E. Yagmurlu, F. Wenzel, and R. Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. *arXiv preprint arXiv:2407.05996*, 2024.
- [60] Y. Liu, P. Ding, T. Jiang, X. Wang, W. Song, M. Lin, H. Zhao, H. Zhang, Z. Zhuang, W. Zhao, S. Huang, J. Shi, and D. Wang. MMaDA-VLA: Large diffusion vision-language-action model with unified multi-modal instruction and generation. *arXiv preprint arXiv:2603.25406*, 2026.

- [61] R. Cai, J. Guo, X. He, P. Jin, J. Li, B. Lin, F. Liu, W. Liu, F. Ma, K. Ma, F. Qiu, H. Qu, Y. Su, Q. Sun, D. Wang, D. Wang, Y. Wang, R. Wu, D. Xiang, Y. Yang, H. Ye, Y. Zhang, and Q. Zhou. Xiaomi-Robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.
- [62] J. Guo, Q. Li, P. Li, Z. Chen, N. Sun, Y. Su, H. Wang, Y. Zhang, X. Li, and H. Liu. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.
- [63] H. He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. *Thinking Machines Lab: Connectionism*, 2025. <https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/>.

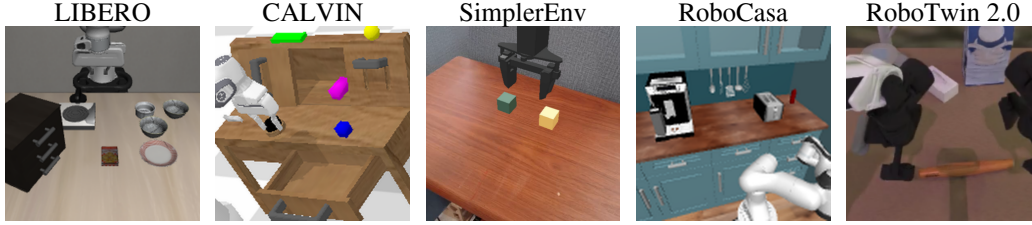


Figure 5: Example rendered observations from the five audited benchmarks, shown only to orient readers to the visual setup. From left to right, LIBERO [2], CALVIN [3], SimplerEnv [4], RoboCasa [5], and RoboTwin 2.0 [6].

A Experiment Details

Figure 5 shows example observations from the five benchmarks we audit.

A.1 Shortcut Solvability

Probe. The probe is a DINOv2 image encoder [34] with a small MLP head, fine-tuned end to end, and it has no diffusion head and no action tokenizer. LIBERO, CALVIN, SimplerEnv, and RoboTwin 2.0 use a ViT-B/14 backbone, while RoboCasa uses the smaller ViT-S/14. Except on SimplerEnv (described below), the probe is never conditioned on language. It is told which task it faces only through a learned per-task embedding. Because the embedding is initialized at random and not taken from any language model, it carries none of the semantic content of a pretrained language embedding. LIBERO, CALVIN, and RoboTwin 2.0 use a multi-task model that takes this embedding, while RoboCasa trains a separate model per task and so needs none. The image inputs follow each benchmark’s standard policy interface, the agentview and wrist cameras with proprioception on LIBERO, the static and gripper cameras on CALVIN, and three camera views at 128×128 on RoboCasa.

Probe tuning and checkpoint selection. We train the probe separately for each LIBERO suite and each CALVIN protocol, sweeping learning rate, augmentation strength, and the training-step budget over a small grid, and we report the best of several checkpoints scored on the suite. Training one model per suite is standard practice, since OpenVLA fine-tunes separately on each task suite [50], MoLA fine-tunes and evaluates per suite [51], and CORAL trains one expert per task [52]. The claim of Section 3 is that a shortcut exists, not that one method beats another, so we tune the probe deliberately to test whether any small probe can reach the score from the same inputs.

Best checkpoint versus a fixed checkpoint. The score we report is the best of several checkpoints, each scored on the reported suite. The benchmark gives no separate validation set, so that suite is also the test set. Keeping the checkpoint that scores best on it is therefore selection on the test set. This is optimistic, but the benchmark allows it. This freedom is a weakness of the benchmark, not of our probe, and any submission could exploit it. We do so here and make it visible. Table 2 puts each best-checkpoint score beside the final-checkpoint score, the score at the last training step with no selection. On LIBERO the two differ by 0.6 to 4.4 percentage points, with a final-checkpoint mean of 95.1% against a best-checkpoint mean of 97.6%. On CALVIN they differ by at most 0.09 average tasks completed, and on D→D the reported score is already the final checkpoint.

LIBERO and CALVIN. On LIBERO we train one probe per suite on the official raw HDF5 demonstrations, evaluate under the official protocol with 50 trials per task and 500 per suite, and report success rate. On CALVIN we train one probe per protocol, condition on a 34-way task id, and evaluate on the official 1000-sequence long-horizon benchmark with episode length 360, reporting average tasks completed. We report full evaluations for all three protocols, D→D, ABC→D, and ABCD→D. The probe scores slightly higher on ABC→D than on D→D, which is surprising because D→D trains and tests on the same scene. The cause is how CALVIN labels its training data.

Table 2: Best-checkpoint and final-checkpoint probe scores. The best-checkpoint column is the best of all checkpoints scored on the reported suite and is what the main table reports. The final-checkpoint column is the score at the last training step, with no checkpoint selection. LIBERO values are success rate over 500 trials per suite. CALVIN values are average tasks completed out of 5 over 1000 sequences.

Benchmark	Suite or protocol	Best checkpoint	Final checkpoint
LIBERO	Spatial	99.0%	98.2%
	Object	100.0%	99.4%
	Goal	98.8%	94.8%
	Long	92.4%	88.0%
CALVIN	D→D	3.123	3.123
	ABC→D	3.242	3.149

CALVIN cuts each demonstration into short language-labeled segments, and a segment often begins partway through the motion. For lifting a block, a segment can start with the gripper already above the block and contain only the final grasp. A probe conditioned on the task id alone then never sees the approach to the block during training and cannot perform it at test time. This artifact lowers the probe’s D→D score in particular, so D→D falls below ABC→D.

RoboCasa. On RoboCasa [5] we follow the RSS24 protocol. We train one probe per task on 50 human demonstrations with AdamW for 1500 epochs. Evaluation uses the RSS24 object-instance split B and the five fixed layout and style pairs (1, 1), (2, 2), (4, 4), (6, 9), and (7, 10), with no camera or texture randomization and 50 rollouts per task. We report the strongest variant, a per-task model with an MLP head, pooled over the 24 RSS24 manipulation tasks, $226/1200 = 18.8\%$. Multi-task and transformer-head variants scored lower.

RoboTwin 2.0. For RoboTwin 2.0, we train the probe DINO+MLP policy following the evaluation protocol adopted by recent methods[53, 54]. Specifically, we train a single multi-task policy using demonstrations from all 50 bimanual tasks. For each task, we use 50 demonstrations collected in the clean environment and 500 demonstrations collected in the randomized environment, resulting in 27,500 demonstrations in total. The clean environment uses a fixed clean background, a fixed table height, and no distractor objects on the table. The randomized environment introduces variations in the background, table height, and lighting, and includes multiple distractor objects on the table. To condition the policy on the target task, we assign each task a learned task embedding and concatenate it with the visual feature vector before feeding them into the MLP action head. For evaluation, we test the probe policy on all tasks under both clean and randomized settings, with 100 rollouts per task and setting, following the same evaluation protocol as prior methods. Table 3 reports the per-task success rates.

SimplerEnv. SimplerEnv evaluates WidowX/Bridge policies, and its standard training source is the real-world BridgeData V2 demonstrations [45]. Unlike LIBERO and CALVIN, Bridge does not draw its instructions from a small fixed set, so the task-id lookup we use on those benchmarks does not apply here. We keep the same DINOv2 ViT-B/14 encoder and MLP head, but the model reads the instruction through a pretrained SigLIP text embedding alongside the image and proprioception. We train it on BridgeData V2 and evaluate on the official WidowX tasks of stack, carrot, spoon, and eggplant. The probe scores 0.0%. A small model trained on the standard Bridge data does not reach the SimplerEnv score, so unlike LIBERO the benchmark shows no shortcut of this kind.

Policy sources for Table 1: OpenVLA [50]; X-VLA [35], with the RoboTwin 2.0 X-VLA cell taken from the protocol-matched comparison reported by ABot-M0 [53], because X-VLA’s own RoboTwin 2.0 result does not train with 50 clean plus 500 randomized demonstrations per task; DeeR-VLA [55]; RoboFlamingo [37]; GR00T-N1 [56]; T-MEE [57]; MoH [58]; MoLA [51]; CORAL_{SimVLA} [52]; MDT-V [59]; MMaDA-VLA [60]; Xiaomi-0 [61]; MotuBrain [54]; X-WAM [62].

Table 3: DINO+MLP probe policy per-task success rates on the RoboTwin 2.0 benchmark.

Task	Clean	Rand.	Task	Clean	Rand.
adjust_bottle	95.0%	90.0%	place_can_basket	50.0%	57.0%
beat_block_hammer	47.0%	42.0%	place_cans_plasticbox	97.0%	97.0%
blocks_ranking_rgb	44.0%	55.0%	place_container_plate	62.0%	51.0%
blocks_ranking_size	36.0%	40.0%	place_dual_shoes	24.0%	27.0%
click_alarmclock	64.0%	68.0%	place_empty_cup	90.0%	87.0%
click_bell	83.0%	81.0%	place_fan	42.0%	43.0%
dump_bin_bigbin	48.0%	50.0%	place_mouse_pad	15.0%	20.0%
grab_roller	100.0%	100.0%	place_object_basket	56.0%	62.0%
handover_block	67.0%	67.0%	place_object_scale	50.0%	47.0%
handover_mic	86.0%	80.0%	place_object_stand	79.0%	76.0%
hanging_mug	8.0%	16.0%	place_phone_stand	58.0%	58.0%
lift_pot	99.0%	100.0%	place_shoe	69.0%	67.0%
move_can_pot	69.0%	61.0%	press_stapler	99.0%	92.0%
move_pillbottle_pad	63.0%	59.0%	put_bottles_dustbin	9.0%	7.0%
move_playingcard_away	83.0%	86.0%	put_object_cabinet	50.0%	36.0%
move_stapler_pad	6.0%	5.0%	rotate_qrcode	42.0%	44.0%
open_laptop	96.0%	95.0%	scan_object	82.0%	77.0%
open_microwave	14.0%	21.0%	shake_bottle	100.0%	100.0%
pick_diverse_bottles	83.0%	68.0%	shake_bottle_horizontally	100.0%	100.0%
pick_dual_bottles	97.0%	93.0%	stack_blocks_three	28.0%	31.0%
place_a2b_left	30.0%	24.0%	stack_blocks_two	62.0%	58.0%
place_a2b_right	25.0%	32.0%	stack_bowls_three	20.0%	25.0%
place_bread_basket	83.0%	80.0%	stack_bowls_two	60.0%	58.0%
place_bread_skillet	80.0%	69.0%	stamp_seal	33.0%	29.0%
place_burger_fries	91.0%	94.0%	turn_switch	46.0%	45.0%
Average	60.4%	59.4%			

A.2 Creeping Overfitting

This appendix gives the SimplerEnv details for Section 5.

Setup. We evaluate four WidowX/Bridge policies on the SimplerEnv stack task: CogACT-Base, InternVLA-M1, X-VLA-WidowX, and Dexbotic (DB-MemVLA). All runs use the standard 60-step limit for this task.

Matched local calibration. The official WidowX stack test is a fixed grid of 24 start states. Our calibration reruns that grid on our hardware, repeating each start state 12 times, for 288 episodes per policy. We report each change as the shift in stack success against this calibration on the same hardware, so hardware and sampling differences drop out and only the change remains.

The three changes. Each change holds the task, scene, camera, robot, and success check fixed, alters one factor, and runs 288 episodes per policy.

- *Reverse language.* The instruction asks for yellow on green instead of the official green on yellow.
- *Stacked support.* The two cubes start on one or two support blocks instead of directly on the table.
- *Random pose and arm.* The cube positions and the initial arm pose are redrawn at random instead of taken from the fixed grid.

These three belong to a larger run of seven conditions (2,016 episodes per policy) released with the paper.

Why the changes stay in distribution. The official test uses one fixed grid, but the BridgeData V2 training demonstrations [45], recorded on a real WidowX robot, are not tied to that grid, so redrawn cube positions and arm poses are ordinary training states. Stacked blocks also appear in training. Across 53,192 demonstrations, “block” appears in 869, “cube” in 751, and “tower” in 282. Both color orders appear, the official green on yellow in 41 demonstrations and the reversed yellow on green in 30. Each change therefore stays close to the training data and is not much less common than the official condition.

Results. Table 4 reports stack success for each policy on the calibration and the three changes.

Table 4: SimplerEnv stack success rate (%) on the matched local calibration and the three in-distribution changes, 288 episodes per policy per column. Some calibration values differ from the numbers these policies report. This is expected, not an error. Several of these policies report under longer, easier episode step limits, while we hold every policy to the same shorter standard step limit.

Policy	Calibration	Reverse language	Stacked support	Random pose + arm
CogACT-Base	20.8	9.7	22.2	10.8
InternVLA-M1	25.3	16.0	20.8	16.0
X-VLA-WidowX	59.7	53.1	31.3	55.6
Dexbotic (DB-MemVLA)	44.8	47.6	27.4	38.9

Confidence intervals. We report two-sided 95% confidence intervals for the distribution-overfitting and sample-overfitting drops, recomputed from the first-hand rollout CSV and JSON outputs. Every drop uses

$$\text{drop} = (\text{matched local calibration}) - (\text{altered or fresh result}),$$

so a positive value means the altered or fresh test scored lower than the matched local calibration and a negative value means it scored higher.

Interval methods. *Binary success rates.* For the SimplerEnv and LIBERO success-rate drops we report a Newcombe–Wilson interval for the difference of two proportions. For a calibration count x_c/n_c and an altered or fresh count x_a/n_a , write $p_c = x_c/n_c$, $p_a = x_a/n_a$, and $\text{drop} = p_c - p_a$. We first form Wilson score intervals $[L_c, U_c]$ and $[L_a, U_a]$ for the two proportions with $z = z_{0.975} \approx 1.95996$. For a proportion $p = x/n$, the Wilson interval is

$$\text{center} = \frac{p + z^2/(2n)}{1 + z^2/n}, \quad \text{half} = \frac{z \sqrt{p(1-p)/n + z^2/(4n^2)}}{1 + z^2/n},$$

$$[L, U] = [\text{center} - \text{half}, \text{center} + \text{half}].$$

The Newcombe interval for $p_c - p_a$ is

$$\text{lower} = \text{drop} - \sqrt{(p_c - L_c)^2 + (U_a - p_a)^2}, \quad \text{upper} = \text{drop} + \sqrt{(U_c - p_c)^2 + (p_a - L_a)^2}.$$

This treats rollout rows as independent Bernoulli outcomes. For SimplerEnv it is a row-level uncertainty summary for the local rollout protocol, and it does not model the fact that the calibration repeats the same 24 official grid instances 12 times.

CALVIN distribution overfitting. The calibration and resampled-pose results share the same canonical 1000-sequence manifest, so we use a paired row-level Wald interval over global sequence indices. With d_i the per-sequence difference in tasks completed,

$$d_i = (\text{calibration count})_i - (\text{resampled count})_i, \quad \text{drop} = \text{mean}_i d_i,$$

$$\text{CI} = \text{drop} \pm z \frac{\text{sd}(d_i)}{\sqrt{1000}}.$$

The same paired calculation gives the 5/5 chain-success drops after converting each sequence to the binary indicator $\mathbf{1}[\text{count} = 5]$.

CALVIN sample overfitting. The calibration uses the official 1000-sequence manifest and the fresh result pools two independent fresh 1000-sequence manifests, so we use an independent two-sample Wald interval over per-sequence tasks-completed counts in $\{0, 1, 2, 3, 4, 5\}$,

$$\text{drop} = \text{mean}(\text{calibration}) - \text{mean}(\text{fresh}),$$

$$\text{SE} = \sqrt{\frac{\text{var}(\text{calibration})}{1000} + \frac{\text{var}(\text{fresh})}{2000}}, \quad \text{CI} = \text{drop} \pm z \text{ SE}.$$

Distribution-overfitting drops. Table 5 reports the four-policy, three-change SimplerEnv stack-task drops in percentage points, and Table 6 reports the CALVIN resampled-pose drops, with average tasks completed (ATC) in tasks out of 5 and the 5/5 chain-success drops in percentage points.

Table 5: SimplerEnv distribution-overfitting drops. Calibration and altered columns are success counts out of 288 episodes per policy. Drop and 95% CI are in percentage points, with a positive drop meaning the altered test scored lower than the matched local calibration.

Policy	Change	Calibration	Altered	Drop	95% CI
CogACT-Base	Reverse language	60/288	28/288	+11.11	[+5.26, +16.95]
CogACT-Base	Stacked support	60/288	64/288	-1.39	[-8.09, +5.33]
CogACT-Base	Random pose + arm	60/288	31/288	+10.07	[+4.13, +15.99]
InternVLA-M1	Reverse language	73/288	46/288	+9.38	[+2.76, +15.91]
InternVLA-M1	Stacked support	73/288	60/288	+4.51	[-2.38, +11.35]
InternVLA-M1	Random pose + arm	73/288	46/288	+9.38	[+2.76, +15.91]
X-VLA-WidowX	Reverse language	172/288	153/288	+6.60	[-1.49, +14.57]
X-VLA-WidowX	Stacked support	172/288	90/288	+28.47	[+20.46, +35.96]
X-VLA-WidowX	Random pose + arm	172/288	160/288	+4.17	[-3.88, +12.14]
Dexbotic (DB-MemVLA)	Reverse language	129/288	137/288	-2.78	[-10.84, +5.33]
Dexbotic (DB-MemVLA)	Stacked support	129/288	79/288	+17.36	[+9.54, +24.89]
Dexbotic (DB-MemVLA)	Random pose + arm	129/288	112/288	+5.90	[-2.14, +13.84]

Table 6: CALVIN distribution-overfitting drops on the shared 1000-sequence manifest. ATC drops are in tasks out of 5; 5/5 chain-success drops are in percentage points. A positive drop means the resampled-pose test scored lower than the calibration.

Policy	Metric	Calibration	Resampled	Drop	95% CI
X-VLA	ATC	4.165	3.138	+1.027	[+0.890, +1.164]
X-VLA	5/5 chain success	709/1000	459/1000	+25.00	[+21.29, +28.71]
GR-1	ATC	3.244	2.495	+0.749	[+0.624, +0.874]
GR-1	5/5 chain success	431/1000	296/1000	+13.50	[+10.23, +16.77]
RoboFlamingo	ATC	2.367	1.869	+0.498	[+0.372, +0.624]
RoboFlamingo	5/5 chain success	209/1000	145/1000	+6.40	[+3.57, +9.23]

Sample-overfitting drops. Table 7 reports the LIBERO fresh-init-state drops in percentage points, and Table 8 reports the CALVIN fresh-sequence drops, where the fresh column pools the two pre-declared fresh 1000-sequence manifests and ATC is in tasks out of 5.

Table 7: LIBERO sample-overfitting drops. Calibration is success counts out of 2000 episodes per policy; the fresh sample redraws the init-state file from the same generator with 10,000 rollouts per policy. Drop and 95% CI are in percentage points.

Policy	Metric	Calibration	Fresh sample	Drop	95% CI
Spatial Forcing	success rate	1956/2000	9718/10000	+0.62	[-0.18, +1.27]
SimVLA	success rate	1946/2000	9760/10000	-0.30	[-1.15, +0.40]
$\pi_{0.5}$ / LeRobot	success rate	1951/2000	9741/10000	+0.14	[-0.69, +0.82]

Table 8: CALVIN sample-overfitting drops. The fresh sample pools two predeclared independent fresh 1000-sequence manifests. ATC is in tasks out of 5, with a positive drop meaning the fresh sample scored lower than the calibration.

Policy	Metric	Calibration	Fresh sample	Drop	95% CI
X-VLA	ATC	4.165	4.1795	-0.0145	$[-0.1287, +0.0997]$
GR-1	ATC	3.244	3.1370	+0.1070	$[-0.0373, +0.2513]$
RoboFlamingo	ATC	2.367	2.2980	+0.0690	$[-0.0677, +0.2057]$

Confidence-interval source artifacts. The intervals were recomputed directly from the per-episode and per-sequence rollout outputs rather than from summary notes. The SimplierEnv drops use the 288-episode fixed-grid calibration and the three altered-condition rollout outputs. The CALVIN drops use the official 1000-sequence aggregates for the calibration and the resampled-pose test, and the two predeclared fresh 1000-sequence aggregates for the sample-overfitting experiment. The LIBERO drops use the official per-suite calibration and the 10,000-rollout fresh-init-state outputs, with a small number of cells taken from documented retry reruns.

A.3 Data-Source Dependence

This appendix gives the SimplierEnv details for Section 6. The diagnostic is intentionally narrow: SimplierEnv evaluates policies in a simulated WidowX setup, while high benchmark scores are often read as evidence that a policy transfers from real BridgeData V2 demonstrations [45] to that simulator. Because SimplierEnv does not restrict training-data sources, we can instead train on scripted demonstrations generated in the same simulator around the official evaluation grid. This is not a claim that scripted demonstrations are comparable to real Bridge demonstrations; it tests whether the score itself identifies transfer across the data-source gap.

Protocol. We use the official SimplierEnv WidowX evaluation grid for four Bridge-style tasks: stack, carrot, spoon, and eggplant. Each task has 24 fixed grid instances, for 96 closed-loop evaluations in total. Training data are generated separately for each task, and evaluation runs on grid ids 0–23 without the training jitter, using the benchmark task definitions, observations, episode step limits, and success checks. We report counts as well as percentages because each task has only 24 trials.

Scripted demonstrations. For each task we collect 120 successful scripted expert demonstrations in simulation, exactly 5 per official grid id. The demonstrations use small object perturbations around the official grid, with 0.0025 m XY jitter and up to 15° yaw jitter, and include terminal open/place padding so the final release behavior is represented in the behavior-cloning targets. The carrot dataset additionally canonicalizes gripper yaw modulo π . The generator uses the public official grid ids only as centers for the jittered demonstration distribution. This deliberately benchmark-adjacent setup tests whether unrestricted training data can avoid, rather than solve, the Bridge-to-SimplierEnv data-source gap.

Policy and training. We train one task-specific DINOv2-S + MLP behavior-cloning policy per task, initialized with a pretrained DINOv2-S visual encoder. Each policy is trained as a separate BC run on that task’s scripted demonstrations. The visual encoder is fine-tuned, image augmentation is disabled, proprioception is included, actions are trained in the SimplierEnv environment-action convention with dataset-standard normalization, and deployment uses 5-step action chunks with replanning every 5 steps. Loader-side terminal-action padding is enabled for training. There is no checkpoint selection. These runs are not an unbiased success-rate estimate: generator fixes were selected to demonstrate the existence of a high-scoring policy on the fixed grid.

Interpretation. The result in Table 9 is a fixed-grid existence result, not an unbiased success-rate estimate and not a controlled comparison to X-VLA. It shows that, when training data is unrestricted, a 91/96 (94.8%) score on the official fixed grid can be achieved by removing the data-source gap rather than crossing it. Therefore the score alone cannot distinguish the two explanations.

Table 9: SimplerEnv WidowX data-source-dependence results. Each task row uses a separate DINOv2-S + MLP policy trained only on 120 scripted demonstrations generated in the same simulator around the official evaluation grid; the final row aggregates all four tasks. Evaluation uses the official 24-instance grid for each task.

Task	Scripted demos	Eval successes	Success rate
Stack	120	24/24	100.0%
Carrot	120	23/24	95.8%
Spoon	120	21/24	87.5%
Eggplant	120	23/24	95.8%
All four	480	91/96	94.8%

B Bitwise Nondeterminism in Policy Rollouts

B.1 Reproduction Targets

The strictest reproduction target is bitwise determinism: the same checkpoint, seed, initial state, wrapper, normalization, action postprocessing, simulator, and evaluation code would produce bitwise-identical states, observations, actions, success labels, and final score. This appendix explains why bitwise determinism should not be used as the sole cross-hardware criterion for independent closed-loop manipulation evaluations. A recent technical discussion of LLM inference nondeterminism emphasizes that bitwise reproducibility depends on the chosen system boundary and numerical implementation details [63]. Here “portable” means applicable across documented but not necessarily identical execution stacks, such as different CPU or GPU hardware.

We separate four increasingly less exacting targets:

1. **Bitwise determinism:** every simulator state, rendered observation, policy input, action, reward, done flag, and success label is bitwise identical at every step.
2. **Per-episode outcome agreement:** per-episode success or failure labels match, even if hidden states or pixels differ.
3. **Aggregate-score agreement:** the final success rate or CALVIN average-tasks-completed (ATC) score matches, possibly without matching the same episodes.
4. **Protocol and statistical reproducibility:** the independent run follows the documented benchmark protocol, records per-episode outcomes when possible, and gives aggregate behavior compatible with finite-rollout uncertainty.

Our paper claims use the last standard, while treating bitwise determinism as a useful diagnostic. This does not relax artifact standards: exact commands, configs, checkpoints, seeds, initial states, wrappers, normalization, postprocessing, success parsing, hardware, driver, CUDA, PyTorch, simulator, and environment metadata are all part of the reproducible object.

B.2 Diagnostic Evidence

Closed-loop evaluation makes bitwise determinism a fragile requirement. A manipulation benchmark does not merely apply a fixed action log to a simulator. The simulator and policy form a discrete-time dynamical system: the simulator state determines rendered images and proprioception; those observations determine the next policy action; the next action changes contacts and future states. Tiny numerical differences from a simulator step or policy forward pass can therefore remain invisible for many steps, or they can be amplified by contact and feedback into different pixels, different actions, and different success labels. Executing a fixed action log open-loop is not sufficient evidence for this question because it removes the policy-simulator feedback loop that benchmark evaluation is meant to measure.

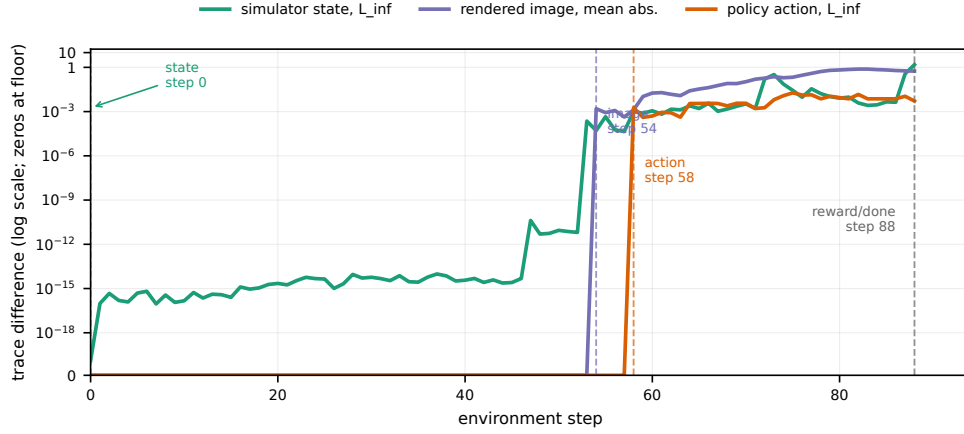


Figure 6: Closed-loop propagation in one LIBERO-Goal task 8 hardware comparison. With policy, seed, initial state, wrapper, and rollout code fixed, changing CPU hardware while keeping the GPU class fixed first changes post-step simulator state at step 0; the difference reaches rendered images at step 54, actions at step 58, and reward/done at step 88. State and action curves use L_∞ distance; the image curve uses mean absolute pixel difference. Zeros are drawn at the log-scale floor.

Figure 6 illustrates the mechanism: the first bitwise difference is tiny and internal to the simulator, but later enters pixels, actions, and the success parser through the feedback loop. In this task, contact instrumentation also localized the growth around contact/constraint dynamics involving the bowl, including a bowl-table contact-mode mismatch during the grasp/lift phase.

Table 10: Diagnostic closed-loop trace-comparison tests. Each row is one CPU or GPU hardware contrast for one benchmark stack. Same-node controls were bitwise identical in the recorded arrays; the goal is not to survey all benchmark stacks, but to show that bitwise determinism can fail even when the benchmark protocol is fixed.

Stack	Hardware contrast	Observed propagation	Bitwise identical?
LIBERO / OpenVLA-OFT	CPU: AMD EPYC 74F3 + RTX A6000 versus Intel Xeon Silver 4210R + RTX A6000	10/10 tasks diverged in simulator state; 5/10 reached pixels and actions.	No
LIBERO / OpenVLA-OFT	GPU: same AMD EPYC 74F3 CPU model, RTX A6000 versus RTX 6000 Ada	Raw and processed actions diverged at step 10, followed by simulator state, image, and reward/done divergence.	No
CALVIN / GR-1	CPU: AMD-CPU node with RTX A6000 versus Intel-CPU node with RTX A6000	0/50 official sequence pairs diverged in this diagnostic sample.	Yes
CALVIN / GR-1	GPU: same AMD EPYC 74F3 CPU model, RTX A6000 versus RTX 6000 Ada	Raw and processed actions diverged at step 0, followed by simulator state and image-hash divergence.	No
SimplerEnv / CogACT	CPU: AMD-CPU node with RTX A6000 versus Intel-CPU node with RTX A6000	Actions and post-step simulator state diverged at step 0; images at step 3; reward/done/success at step 38.	No
SimplerEnv / CogACT	GPU: same AMD EPYC 74F3 CPU model, RTX A6000 versus RTX 6000 Ada	Actions and simulator state diverged from step 0, followed by image divergence; both traces still ended unsuccessfully.	No
RoboCasa / StarVLA	CPU: AMD-CPU node with RTX A6000 versus Intel-CPU node with RTX A6000	Simulator and contact traces diverged immediately; image/action divergence followed at step 4 and reward/success at step 19.	No
RoboCasa / StarVLA	GPU: same AMD EPYC 74F3 CPU model, RTX A6000 versus RTX 6000 Ada	Actions, contact traces, simulator state, and images diverged from the first steps; reward/success diverged at step 18.	No

The LIBERO CPU contrast is the clearest positive case: with GPU class held fixed as RTX A6000, changing CPU hardware broke bitwise determinism and sometimes changed the final outcome. The same-node controls and the CALVIN CPU screen bound the claim in the other direction. Bitwise

determinism can hold; it is just not a property that should be assumed across independent hardware stacks.

B.3 Implications for This Paper

This distinction matters for paper sections that rely on rerunning documented benchmark protocols. For statistical significance, the relevant object is the per-instance success/failure table under a documented protocol, because that is what determines the paired uncertainty in Section 4. Recovering the exact decimal in a published aggregate table would not by itself establish a significant improvement. Conversely, a small aggregate mismatch does not by itself refute reproduction when protocol fidelity is established and the discrepancy is statistically compatible.

For shortcut solvability, the relevant claim is also protocol-level: under official evaluation protocols, simple DINO+MLP policies with task-ID inputs reach high benchmark scores on LIBERO and CALVIN. The inference failure is that those benchmark scores do not identify language understanding or large-scale VLA pretraining.

The practical implication is to strengthen, not relax, the artifact standard. Authors should release enough information for a reader to audit protocol fidelity and to interpret statistically compatible discrepancies: checkpoint references, code revisions, commands, configs, seeds, initial states, wrappers, normalization, action postprocessing, success parsing, CPU and GPU metadata, driver, CUDA, PyTorch, simulator versions, environment identifiers, and per-episode outcome files. Exact bit-wise determinism is valuable when it is available, but it should not be required as a cross-hardware pass/fail criterion when protocol fidelity, per-episode outcomes, and statistical uncertainty are reported.

C Top-Line Cutoffs for Bounded-Score Wald Tests

This appendix derives top-line-only cutoffs for bounded-score benchmark comparisons. The primary derivation is for paired evaluations, where both policies are evaluated on the same benchmark instances. We then show that the paired cutoffs remain conservative for independently sampled evaluations with the same task structure and the same number of samples per task.

Binary success/failure benchmarks correspond to $R = 1$. CALVIN corresponds to $R = 5$, $T = 1$, and $S = 1000$.

C.1 Paired Task-Stratified Wald Rule

We compare two policies on a fixed benchmark with T tasks and S paired samples per task. The tasks are treated as fixed strata, and the paired samples within each task are assumed independent. Unstratified benchmarks are represented by setting $T = 1$. Assume $S > 1$ and

$$Y_{A,t,i}, Y_{B,t,i} \in \{0, 1, \dots, R\}.$$

For task t and sample i , define the paired score difference

$$\delta_{t,i} = Y_{B,t,i} - Y_{A,t,i} \in \{-R, \dots, R\}.$$

Let

$$\Delta_t = \mathbb{E}[\delta_{t,i}] = \mu_{B,t} - \mu_{A,t}, \quad \sigma_t^2 = \text{Var}(\delta_{t,i}),$$

and define the fixed-task estimand

$$\Delta = \frac{1}{T} \sum_{t=1}^T \Delta_t.$$

Because every task has the same number S of samples, the corresponding empirical estimator is equal to the difference in top-line empirical metrics:

$$\hat{\Delta}_t = \frac{1}{S} \sum_{i=1}^S \delta_{t,i}, \quad \hat{\Delta} = \frac{1}{T} \sum_{t=1}^T \hat{\Delta}_t.$$

For fixed T and $S \rightarrow \infty$, the central limit theorem gives

$$\frac{\hat{\Delta} - \Delta}{\sqrt{\sum_{t=1}^T \frac{\sigma_t^2}{T^2 S}}} \xrightarrow{d} \mathcal{N}(0, 1),$$

provided the limiting variance is nonzero. For finite S , we use the resulting Wald procedure as a nominal large-sample decision rule.

Define

$$d_t = \sum_{i=1}^S \delta_{t,i}, \quad s_t = \sum_{i=1}^S \delta_{t,i}^2.$$

The unbiased sample variance of the paired differences on task t is

$$\hat{\sigma}_t^2 = \frac{1}{S-1} \left(s_t - \frac{d_t^2}{S} \right).$$

Therefore

$$\hat{V} = \sum_{t=1}^T \frac{\hat{\sigma}_t^2}{T^2 S} = \frac{1}{T^2 S(S-1)} \sum_{t=1}^T \left(s_t - \frac{d_t^2}{S} \right),$$

and, when $\hat{V} > 0$,

$$Z = \frac{\hat{\Delta}}{\sqrt{\hat{V}}} = \frac{\sum_{t=1}^T d_t}{\sqrt{\frac{S}{S-1} \sum_{t=1}^T \left(s_t - \frac{d_t^2}{S} \right)}}.$$

For the finite-sample decision rule, we use the following convention when $\hat{V} = 0$:

$$Z_* = \begin{cases} \hat{\Delta}/\sqrt{\hat{V}}, & \hat{V} > 0, \\ +\infty, & \hat{V} = 0 \text{ and } \hat{\Delta} > 0, \\ 0, & \hat{V} = 0 \text{ and } \hat{\Delta} = 0, \\ -\infty, & \hat{V} = 0 \text{ and } \hat{\Delta} < 0. \end{cases}$$

For testing

$$H_0 : \Delta \leq 0 \quad \text{against} \quad H_1 : \Delta > 0,$$

the nominal one-sided Wald rule rejects when

$$Z_* > z_{1-\alpha}.$$

Throughout the top-line bounds below we assume $0 < \alpha < 1/2$, so that $z_{1-\alpha} > 0$.

For binary success/failure outcomes, $R = 1$ and $\delta_{t,i}^2 = 1$ exactly when policies A and B recorded different outcomes. Thus s_t is the task-level discordance count, and the statistic reduces to the usual paired stratified Wald statistic for binary outcomes.

C.2 Top-Line Count Notation

Suppose only the top-line empirical metrics are reported. Let

$$N = TS, \quad A = N\hat{\mu}_A, \quad B = N\hat{\mu}_B, \quad L = B - A,$$

and assume in this subsection that

$$A, B \in \{0, \dots, RN\}, \quad B > A.$$

The observed empirical gap is

$$g = \hat{\mu}_B - \hat{\mu}_A = \frac{L}{N}.$$

For fixed A , the feasible positive count gaps are

$$\mathcal{L}_A = \{1, \dots, RN - A\}.$$

For each $L \in \mathcal{L}_A$, the corresponding top-line count for policy B is $B = A + L$.

The reported top-line counts imply

$$\sum_{t=1}^T d_t = L.$$

The paired Wald denominator depends on the paired outcome table only through

$$Q = \sum_{t=1}^T \left(s_t - \frac{d_t^2}{S} \right).$$

Define

$$c_\alpha = z_{1-\alpha} \sqrt{\frac{S}{S-1}}.$$

For fixed reported counts A and $B = A + L$, the numerator of the Wald statistic is fixed at L . Since $L > 0$, the zero-variance convention above implies that a compatible paired outcome table rejects if and only if

$$L > c_\alpha \sqrt{Q}.$$

This equivalence includes the case $Q = 0$, in which the displayed inequality reduces to $L > 0$.

C.3 Exact Lower Envelope for Paired Evaluations

We first minimize Q over all paired outcome tables compatible with the top-line counts. For any integer x , write

$$x = qS + \rho_S(x), \quad q \in \mathbb{Z}, \quad \rho_S(x) \in \{0, \dots, S-1\}.$$

For a fixed task total d_t , the integer differences $\delta_{t,1}, \dots, \delta_{t,S}$ minimize $\sum_i \delta_{t,i}^2$, subject to $\sum_i \delta_{t,i} = d_t$, by being as equal as possible. Hence every compatible paired outcome table satisfies

$$s_t - \frac{d_t^2}{S} \geq \rho_S(d_t) - \frac{\rho_S(d_t)^2}{S}.$$

Let

$$\psi(r) = r - \frac{r^2}{S}, \quad 0 \leq r < S.$$

For $x, y \in \{0, \dots, S-1\}$,

$$\psi(x) + \psi(y) \geq \psi((x+y) \bmod S).$$

Indeed, if $x + y < S$, the difference is $2xy/S$; if $x + y \geq S$, the difference is $2(S-x)(S-y)/S$. Repeatedly combining residuals across tasks therefore gives a lower bound depending only on the total count gap L . Write

$$L = qS + r, \quad q = \left\lfloor \frac{L}{S} \right\rfloor, \quad 0 \leq r < S.$$

Then every compatible paired outcome table satisfies

$$Q \geq Q_{\text{lo}}(L; T, S),$$

where

$$Q_{\text{lo}}(L; T, S) = r - \frac{r^2}{S}.$$

This lower bound is sharp. Choose integers $c_1, \dots, c_T \in \{0, \dots, R\}$ such that

$$\sum_{t=1}^T c_t = q.$$

If $r > 0$, then $q < RT$, so choose a task t_0 with $c_{t_0} < R$. Define paired differences by

$$\delta_{t,i} = c_t \quad \text{for all } t, i,$$

and, when $r > 0$, add one additional unit to r samples in task t_0 :

$$\delta_{t_0,i} = c_{t_0} + 1 \quad \text{for } i = 1, \dots, r.$$

The total difference is L , and all task-level contributions to Q are zero except the residual task, whose contribution is

$$r - \frac{r^2}{S}.$$

Thus the constructed difference table attains $Q_{\text{lo}}(L; T, S)$.

It remains to realize the specified top-line count A . The total remaining capacity under the constructed nonnegative differences is

$$\sum_{t=1}^T \sum_{i=1}^S (R - \delta_{t,i}) = RN - L \geq A,$$

because $B = A + L \leq RN$. Choose integer baseline scores $X_{t,i} \in \{0, \dots, R - \delta_{t,i}\}$ summing to A , and set

$$Y_{A,t,i} = X_{t,i}, \quad Y_{B,t,i} = X_{t,i} + \delta_{t,i}.$$

This gives a compatible paired outcome table with

$$Q = Q_{\text{lo}}(L; T, S).$$

Therefore Q_{lo} is the exact minimum of Q over compatible paired outcome tables.

Consequently, rejection is impossible whenever

$$L \leq c_\alpha \sqrt{Q_{\text{lo}}(L; T, S)}.$$

Conversely, if the inequality is reversed, then the sharpness construction above gives a compatible paired outcome table that rejects.

The count-scale rejection-feasibility cutoff is

$$L_\exists(A; T, S, R, \alpha) = \min \left\{ L \in \mathcal{L}_A : L > c_\alpha \sqrt{Q_{\text{lo}}(L; T, S)} \right\},$$

with the convention that the minimum is $+\infty$ if the set is empty. The normalized cutoff is

$$\delta_\exists(\hat{\mu}_A; T, S, R, \alpha) = \frac{L_\exists(A; T, S, R, \alpha)}{N}.$$

Thus

$$g < \delta_\exists(\hat{\mu}_A; T, S, R, \alpha)$$

implies that rejection is impossible from the reported top-line metrics.

Since

$$Q_{\text{lo}}(L; T, S) = L - \frac{L^2}{S} \quad \text{for } 1 \leq L \leq S,$$

the first count gap satisfying the feasibility inequality is

$$\ell_\exists^*(S, \alpha) = 1 + \left\lfloor \frac{z_{1-\alpha}^2 S}{S - 1 + z_{1-\alpha}^2} \right\rfloor.$$

Therefore

$$L_\exists(A; T, S, R, \alpha) = \begin{cases} \ell_\exists^*(S, \alpha), & \text{if } \ell_\exists^*(S, \alpha) \leq RN - A, \\ +\infty, & \text{otherwise.} \end{cases}$$

C.4 Exact Upper Envelope for Paired Evaluations

We next maximize Q over all paired outcome tables compatible with the top-line counts. For one task, suppose the task has total score a for policy A and total score b for policy B . Define

$$M_{\max}^{(R)}(a, b; S) = \max_{\{n_{uv}\}} \sum_{u=0}^R \sum_{v=0}^R n_{uv} (v - u)^2$$

subject to

$$\begin{aligned} n_{uv} &\in \mathbb{Z}_{\geq 0}, \\ \sum_{u=0}^R \sum_{v=0}^R n_{uv} &= S, \\ \sum_{u=0}^R \sum_{v=0}^R u n_{uv} &= a, \quad \sum_{u=0}^R \sum_{v=0}^R v n_{uv} = b. \end{aligned}$$

Here n_{uv} is the number of samples in the task on which policy A receives score u and policy B receives score v . The corresponding maximum task-level contribution to Q , conditional on (a, b) , is

$$q_{\max}^{(R)}(a, b; S) = M_{\max}^{(R)}(a, b; S) - \frac{(b - a)^2}{S}.$$

The integer program above is the cleanest definition. The implementation uses the following equivalent closed form. Assume $a \leq b$, and set

$$D = b - a.$$

For $x \geq 0$, write $x = q_x R + r_x$, with $q_x \in \mathbb{Z}_{\geq 0}$ and $0 \leq r_x < R$, and define

$$\Pi_R(x) = q_x R^2 + r_x^2, \quad \nu_R(x) = \begin{cases} 0, & x = 0, \\ \lceil x/R \rceil, & x > 0. \end{cases}$$

Let j denote the total negative movement, so the total positive movement is $D + j$. Feasibility requires

$$0 \leq j \leq a, \quad 0 \leq j \leq RS - b,$$

and the positive and negative movements must fit on disjoint samples:

$$\nu_R(D + j) + \nu_R(j) \leq S.$$

For fixed j , the sum of squared differences is maximized by packing the positive and negative movements into as few samples as possible, with each sample carrying movement of magnitude at most R . Therefore

$$M_{\max}^{(R)}(a, b; S) = \max_{j \in \mathcal{J}(a, b)} \{\Pi_R(D + j) + \Pi_R(j)\},$$

where

$$\mathcal{J}(a, b) = \{j \in \mathbb{Z}_{\geq 0} : j \leq a, j \leq RS - b, \nu_R(D + j) + \nu_R(j) \leq S\}.$$

The objective is nondecreasing in j , so the maximum is attained at the largest feasible j . The case $a > b$ follows by symmetry.

For binary success/failure outcomes, $R = 1$, this reduces to

$$q_{\max}^{(1)}(a, b; S) = \min(a + b, 2S - a - b) - \frac{(b - a)^2}{S}.$$

Here $\min(a + b, 2S - a - b)$ is the largest possible number of discordant samples given the two binary task totals.

We allocate the marginal score totals A and $B = A + L$ across tasks by dynamic programming. Let $F_j(u, v)$ be the largest attainable value of Q after j tasks with u total score for policy A and v total score for policy B . Initialize

$$F_0(0, 0) = 0, \quad F_0(u, v) = -\infty \quad \text{for } (u, v) \neq (0, 0).$$

For $j = 0, \dots, T - 1$, update

$$F_{j+1}(u + a', v + b') = \max \left\{ F_{j+1}(u + a', v + b'), F_j(u, v) + q_{\max}^{(R)}(a', b'; S) \right\}$$

over every finite state $F_j(u, v)$ and every

$$0 \leq a' \leq RS, \quad 0 \leq b' \leq RS.$$

Define

$$Q_{\text{hi}}(A, L; T, S, R) = F_T(A, A + L).$$

Because the dynamic program enumerates every allocation of the two marginal top-line counts across tasks, and because $q_{\max}^{(R)}$ is the exact task-level maximum for each allocation, Q_{hi} is the exact maximum of Q over all paired outcome tables compatible with top-line counts A and $B = A + L$. Thus every compatible paired outcome table satisfies

$$Q \leq Q_{\text{hi}}(A, L; T, S, R),$$

and this upper bound is attained.

Therefore rejection is guaranteed whenever

$$L > c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)}.$$

Conversely, if

$$L \leq c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)},$$

then the sharpness of Q_{hi} gives a compatible paired outcome table that does not reject.

The count-scale rejection-guarantee cutoff used for scalar summaries is the suffix cutoff

$$L_{\forall}(A; T, S, R, \alpha) = \min \left\{ \ell \in \mathcal{L}_A : \begin{array}{l} \text{for every } L \in \mathcal{L}_A \text{ with } L \geq \ell, \\ L > c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)} \end{array} \right\},$$

with the convention that the minimum is $+\infty$ if the set is empty. The normalized cutoff is

$$\delta_{\forall}(\hat{\mu}_A; T, S, R, \alpha) = \frac{L_{\forall}(A; T, S, R, \alpha)}{N}.$$

Thus

$$g \geq \delta_{\forall}(\hat{\mu}_A; T, S, R, \alpha)$$

implies that every paired outcome table compatible with the reported top-line metrics rejects under the paired bounded-score Wald rule. For a particular observed gap L/N , the sharper pointwise guarantee condition is

$$L > c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)}.$$

C.5 Sharp Paired Classification

For exact paired evaluations, the two envelopes above give a sharp classification for each fixed count pair $(A, B = A + L)$:

$$\begin{aligned} L &\leq c_\alpha \sqrt{Q_{\text{lo}}(L; T, S)} && \iff \text{no compatible paired outcome table rejects,} \\ L &> c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)} && \iff \text{every compatible paired outcome table rejects,} \\ c_\alpha \sqrt{Q_{\text{lo}}(L; T, S)} < L &\leq c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)} && \iff \begin{array}{l} \text{some compatible paired outcome tables reject,} \\ \text{and some compatible paired outcome tables do not reject.} \end{array} \end{aligned}$$

The middle line is the sharp top-line inconclusive case for paired evaluations.

C.6 Independently Sampled Evaluations

Some published comparisons do not use the same paired benchmark instances for both policies. Instead, each method is evaluated on independently sampled task-conditioned instances, with the same T tasks and the same number S of samples per task. In this setting, the paired difference statistic is not the appropriate reference procedure. The corresponding stratified two-sample Wald statistic uses separate within-task variance estimates for the two policies.

Let

$$a_t = \sum_{i=1}^S Y_{A,t,i}, \quad b_t = \sum_{i=1}^S Y_{B,t,i},$$

and

$$u_t = \sum_{i=1}^S Y_{A,t,i}^2, \quad v_t = \sum_{i=1}^S Y_{B,t,i}^2.$$

The two-sample denominator is determined by

$$Q_{\text{ind}} = \sum_{t=1}^T \left[u_t - \frac{a_t^2}{S} + v_t - \frac{b_t^2}{S} \right].$$

The count-scale stratified two-sample Wald statistic is

$$Z_{\text{ind}} = \frac{L}{\sqrt{\frac{S}{S-1} Q_{\text{ind}}}},$$

with the same zero-variance convention as above.

For binary success/failure outcomes, this simplifies to

$$Q_{\text{ind}} = \sum_{t=1}^T \left[a_t - \frac{a_t^2}{S} + b_t - \frac{b_t^2}{S} \right].$$

The paired envelopes are conservative for this independently sampled setting. For every independently sampled outcome table with top-line counts A and $B = A + L$,

$$Q_{\text{lo}}(L; T, S) \leq Q_{\text{ind}} \leq Q_{\text{hi}}(A, L; T, S, R).$$

For the lower inequality, fix a task t . Let X and Y be independent draws from the empirical score distributions of policies A and B on that task. Then $Y - X$ is integer-valued,

$$\mathbb{E}[Y - X] = \frac{b_t - a_t}{S},$$

and

$$S \text{Var}(Y - X) = u_t - \frac{a_t^2}{S} + v_t - \frac{b_t^2}{S}.$$

Among integer-valued random variables with a fixed mean, the variance is minimized by placing mass on the two adjacent integers around the mean. Therefore, with $\rho_t = \rho_S(b_t - a_t)$,

$$u_t - \frac{a_t^2}{S} + v_t - \frac{b_t^2}{S} \geq \rho_t - \frac{\rho_t^2}{S}.$$

Combining residuals across tasks as in the paired lower-envelope proof gives

$$Q_{\text{ind}} \geq Q_{\text{lo}}(L; T, S).$$

For the upper inequality, again fix a task t . Center the two empirical score lists by subtracting their task means. Over a uniformly random pairing of the A -scores and B -scores within the task, the

sample covariance has expectation zero. Hence there exists at least one pairing with covariance $C_t \leq 0$. For that pairing, the paired task contribution satisfies

$$Q_{\text{pair},t} = \left(u_t - \frac{a_t^2}{S} + v_t - \frac{b_t^2}{S} \right) - 2C_t \geq u_t - \frac{a_t^2}{S} + v_t - \frac{b_t^2}{S}.$$

Applying this argument independently within each task gives a compatible paired outcome table whose Q is at least Q_{ind} . Since $Q_{\text{hi}}(A, L; T, S, R)$ is the maximum paired value, it follows that

$$Q_{\text{ind}} \leq Q_{\text{hi}}(A, L; T, S, R).$$

Thus the paired cutoffs remain valid as conservative top-line cutoffs for independently sampled evaluations:

$$L \leq c_\alpha \sqrt{Q_{\text{lo}}(L; T, S)} \implies \text{no compatible independently sampled outcome table rejects,}$$

and

$$L > c_\alpha \sqrt{Q_{\text{hi}}(A, L; T, S, R)} \implies \text{every compatible independently sampled outcome table rejects.}$$

However, the intermediate region is not sharp for independently sampled evaluations. It means only that the paired top-line bounds do not resolve the corresponding stratified two-sample Wald test. Sharper two-sample top-line bounds could classify some cases inside this region.

C.7 Non-Realizable Metric Rounding

The derivations above assume that the reported metrics determine exact integer score totals. Some methods, for example ones with non-deterministic policies, may report an aggregate score averaged over multiple runs over the test set with different seeds. Such reported scores may not be realizable from a single run over the test set. Address this by applying a convention that rounds the reported score to the nearest score realizable with a standard pass over the test set, and then treating it as we would the standard case.

D Citation Tracker Methodology

We use a citation tracker repository to derive both the count of new arXiv papers reporting numbers on a given benchmark in a given month (the “79 new arXiv papers” figure in the introduction) and the per-benchmark cumulative counts in Figure 2. Per-paper evidence and the full row-level audit are included in the anonymized supplementary zip submitted through OpenReview. The snapshot we report on was taken on 2026-05-21.

D.1 Sources

Candidates were harvested from Semantic Scholar (citation graph and term search), OpenAlex (citation graph, term search, DOI metadata), CORE (term and full-text search), the arXiv API (term search and metadata), OpenReview, Europe PMC, OpenCitations / COCI (citation-graph spot checks), OpenAIRE / Scholixplorer (DOI / arXiv linkages), Crossref (DOI metadata and references), DataCite (DOI metadata, including arXiv DOIs), DBLP, NeurIPS proceedings pages, and project, code repository, and documentation pages. We did not use Google Scholar / Publish or Perish exports, Lens.org, Scopus, Web of Science, or Dimensions. Any paper indexed only by an excluded source is missed; the reported counts are therefore lower bounds.

D.2 Deduplication

Candidates are deduplicated by DOI, arXiv ID, Semantic Scholar paper ID, OpenAlex work ID, and normalized title. A candidate surfaced by multiple sources is kept once; all source-level matches are preserved in a Found via column for row-level auditability.

D.3 Per-Paper Classification

Every candidate carries a *usage* classification per benchmark:

- **reports-results**: the paper reports at least one numeric success-rate (or task-completion) value for the benchmark, with the specific protocol or suite identified. This requires row-level full-text audit, not just a citation edge or a term match.
- **citation-only**: the paper cites the benchmark seed paper but reports no numeric results.
- **term-only-not-counted**: the paper matches a term search for the benchmark name but lacks a confirmed citation edge or reported result. Retained for source-coverage transparency; not counted.
- **uses-data-no-results**: the paper trains on or otherwise reuses benchmark data without running its standard evaluation.
- **unclear/full-text-unavailable**: full text could not be retrieved at audit time.
- **seed-work**: the benchmark seed paper itself.

Only **reports-results** rows are counted as “papers that reported numbers on the benchmark.” The promotion rule is one-directional: we do not infer reported results from a citation or a name mention alone, and collision-prone names (“LIBERO,” “CALVIN,” “Habitat,” and similar short tokens) stay at **term-only-not-counted** until row-level evidence promotes them.

D.4 Date Convention

Monthly bucketing uses the first arXiv posting date (v1), not the venue publication date. We chose this because the venue date can lag the public claim by months to years, and benchmark adoption tracks the public claim. Rows without an arXiv ID are excluded from monthly bucketing.

D.5 Reproducing the Headline Count

The introduction’s “79 new arXiv papers reported numbers on LIBERO in March 2026” is the number of rows in the LIBERO tracker for which `First arXiv date` begins with 2026-03 and LIBERO usage equals **reports-results**. At the 2026-05-21 snapshot, the LIBERO tracker contained 1,053 candidate rows, of which 551 are classified **reports-results**. March 2026 contributes 79 of those, drawn from 117 March 2026 candidates overall: the remaining 38 are predominantly **citation-only**, with a smaller number in the other classes.

D.6 Figure 2

Figure 2 applies the same filter to CALVIN, LIBERO, SimplerEnv, RoboCasa, and RoboTwin 2.0, bucketed by first arXiv date from July 2023 through March 2026. Each benchmark uses an independent tracker, populated by the same source pipeline and classified under an analogous (`benchmark`) usage column. A paper that reports results on multiple benchmarks is counted once per benchmark.