# vla-eval: A Unified Evaluation Harness for Vision-Language-Action Models

Suhwan Choi[1*], Yunsung Lee[1], Yubeen Park[1], Chris Dongjoo Kim[2], Ranjay Krishna[2], Dieter Fox[2], Youngjae Yu[3]
[1]MAUM.AI [2]Allen Institute for AI (AI2) [3]Seoul National University
[*]Primary author

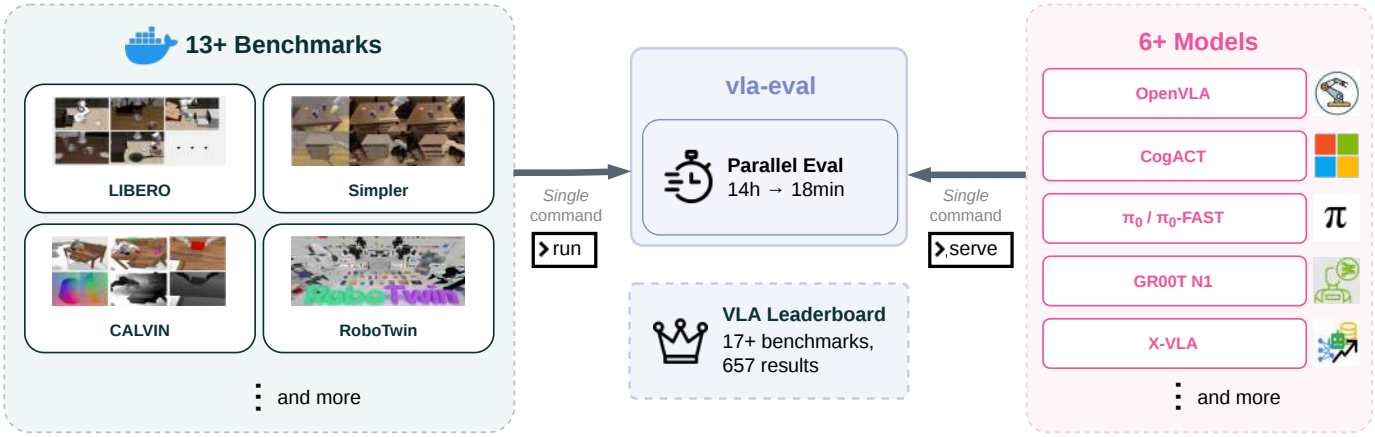


Fig. 1. Overview of `vla-eval`: 14 benchmarks (3 with full cross-codebase reproduction validation) and six model servers each integrate once, requiring no per-benchmark dependency setup or manual asset installation, and connect through two commands (`run` and `serve`). The framework provides parallel evaluation (up to 47× speedup on LIBERO: 14h → 18min) and a VLA leaderboard aggregating 657 results across 17 benchmarks.

***Abstract*—Vision-Language-Action (VLA) models are increasingly evaluated across multiple simulation benchmarks, yet adding each benchmark to an evaluation pipeline requires resolving incompatible dependencies, matching underspecified evaluation protocols, and reverse-engineering undocumented preprocessing. This burden scales with the number of models and benchmarks, making comprehensive evaluation impractical for most teams. We present `vla-eval`, an open-source evaluation harness that eliminates this per-benchmark cost by decoupling model inference from benchmark execution through a WebSocket+msgpack protocol with Docker-based environment isolation. Models integrate once by implementing a single `predict()` method; benchmarks integrate once via a four-method interface; the full cross-evaluation matrix works automatically. The framework supports 14 simulation benchmarks and six model servers. Parallel evaluation via episode sharding and batch inference achieves up to 47× wall-clock speedup, completing 2,000 LIBERO episodes in ∼18 minutes. To validate the framework, we reproduce published scores across six VLA codebases and three benchmarks, documenting previously undocumented pitfalls. We additionally release a VLA leaderboard aggregating 657 published results across 17 benchmarks. Framework, evaluation configs, and all reproduction results are publicly available.[1,2]**

***Index Terms*—VLA evaluation, robot manipulation, benchmark, reproducibility**

## I. Introduction

Recent Vision-Language-Action (VLA) models increasingly target multiple simulation benchmarks to demonstrate generalization across environments and embodiments [1]–[4]. However, adding even a single benchmark to an evaluation pipeline demands substantial engineering effort. Each benchmark ships its own simulator, Python runtime, and asset requirements—LIBERO [5] requires Python 3.8 with robosuite, ManiSkill2 [6] requires Python 3.10 with SAPIEN, CALVIN [7] requires Python 3.8 with PyBullet—and no single environment can satisfy all constraints simultaneously. Beyond dependency resolution, evaluation protocols are frequently underspecified: seeds, episode counts, and preprocessing details are omitted from papers, and a single undocumented parameter can shift success rates by up to 55 percentage points (Section III). Correctly integrating one benchmark therefore requires not just environment setup but painstaking comparison against reference implementations.

This per-benchmark cost scales linearly: evaluating on $M$ benchmarks means repeating the process $M$ times, independently for each of $N$ models—an $O(N \times M)$ integration burden. For small teams, comprehensive multi-benchmark evaluation is impractical.

We present `vla-eval` (Fig. 1), a unified evaluation harness that eliminates per-benchmark integration cost. Following the decoupled design of `lm-evaluation-harness` [8] for language models, `vla-eval` isolates each benchmark inside a Docker container and connects it to model servers via a WebSocket+msgpack protocol. *Models integrate once, benchmarks integrate once, and the full $N \times M$ cross-evaluation*

[1]https://github.com/allenai/vla-evaluation-harness
[2]https://allenai.github.io/vla-evaluation-harness/leaderboard

```python
class OpenVLAServer(PredictModelServer):
  def __init__(self, model_path, **kw):
    super().__init__(**kw)
    self.model_path = model_path
    self._model = self._proc = None

  def _load_model(self):
    if self._model is not None:
      return
    self._proc = AutoProcessor.from_pretrained(
        self.model_path, trust_remote_code=True)
    self._model = AutoModelForVision2Seq \
        .from_pretrained(self.model_path,
          torch_dtype=torch.bfloat16,
          trust_remote_code=True).to("cuda")

  def predict(self, obs, ctx):
    self._load_model()
    img = Image.fromarray(
        next(iter(obs["images"].values())))
    prompt = f"In: What action should the robot" \
        f" take to {obs['task_description']}?\nOut:"
    inp = self._proc(prompt, img)
        .to("cuda", dtype=torch.bfloat16)
    act = self._model.predict_action(**inp)
    return {"actions": act}
```

Listing 1. OpenVLA model server (simplified).

*matrix works automatically*, reducing integration effort from $O(N \times M)$ to $O(N + M)$. Our contributions are:

- An open-source evaluation harness supporting 14 benchmarks and six model servers with Docker-based isolation and a WebSocket+msgpack protocol;
- Validation across six VLA codebases and three benchmarks, reproducing published scores and documenting pitfalls where a single undocumented parameter shifts success rates by up to 55 percentage points;
- A model-agnostic parallel evaluation methodology (episode sharding + batch inference) achieving up to 47× speedup, where the bottleneck is environment step rate rather than model inference;
- A VLA leaderboard with canonical protocol definitions, aggregating 657 published results across 17 benchmarks.

## II. Framework Design

### A. Architecture

`vla-eval` separates model inference from benchmark execution via a client-server architecture using WebSocket with msgpack binary serialization. Each message carries a type (`observation`, `action`, `episode_start/end`), a benchmark-specific payload, a sequence number, and a timestamp.

**Model servers** extend `PredictModelServer`, which provides a blocking `predict(obs, ctx)` method (typically ∼50 lines), automatic action chunking, and optional batched inference via `max_batch_size`. Listing 1 shows the complete OpenVLA integration.

**Dependency isolation.** Each model server declares dependencies via PEP 723 inline metadata; `vla-eval serve` launches it through `uv run`, creating an isolated environment automatically. Conflicting dependencies (e.g., CogACT pinning `transformers==4.40.1` vs. X-VLA [3] requiring `transformers>=4.44`) coexist without interference, mirroring the Docker-based isolation used for benchmarks.

**Benchmarks** follow the same pattern: integrators implement four methods (`reset`, `step`, `make_obs`, `get_step_result`) inside a dedicated Docker image with pinned dependencies.

**Declarative configs.** Two YAML configs (benchmark + model server) drive each evaluation. We publish all Docker images to `ghcr.io` with versioned tags and bundle all required assets (scene files, textures, robot descriptions), eliminating the ad-hoc asset installation that each benchmark otherwise requires. A complete evaluation requires only two commands: `vla-eval serve` and `vla-eval run`. Every run produces a structured JSON result file recording the harness version, benchmark configuration, and per-episode metrics, enabling exact reproduction.

TABLE I
Supported benchmarks. Docker = compressed image size; Act. = action space dimensionality; St. = status (C = cross-codebase reproduction verified, I = integrated but not yet cross-validated).

| Benchmark | Docker | Act. | St. |
|---|---|---|---|
| SimplerEnv [9] | 4.9 GB | 7D | C |
| LIBERO [5] | 6.0 GB | 7D | C |
| CALVIN [7] | 9.5 GB | 7D | C |
| RLBench [10] | 4.7 GB | 8D | I |
| LIBERO-Pro [11] | 6.2 GB | 7D | I |
| RoboCerebra [12] | 6.3 GB | 7D | I |
| ManiSkill2 [6] | 9.8 GB | 7D | I |
| Kinetix [13] | 10.0 GB | 6D | I |
| MIKASA-Robo [14] | 10.1 GB | 8D | I |
| LIBERO-Mem [15] | 11.3 GB | 7D | I |
| RoboMME [16] | 17.0 GB | 8D | I |
| VLABench [17] | 17.7 GB | 7D | I |
| RoboTwin 2.0 [18] | 28.6 GB | 14D | I |
| RoboCasa [19] | 35.6 GB | 7D | I |

### B. Supported Benchmarks and Models

Table I lists all 14 supported benchmarks with action spaces from 6D to 14D and Docker images from 4.7 to 35.6 GB. Model servers are implemented for six models: CogACT [20], OpenVLA [21], OpenVLA-OFT [22], $\pi_0$ [23]/$\pi_0$-FAST [24], GR00T N1 [1], and X-VLA [3].

### C. Parallel Evaluation

Environment parallelism uses episode sharding across $K$ Docker containers; inference parallelism uses batched forward passes. We tune parallelism via a demand/supply methodology (Fig. 2): $\lambda(K)$ measures environment throughput as a function of shards, $\mu(B)$ measures model throughput as a function of batch size, and the operating point satisfies $\lambda(K) < 0.8 \cdot \mu(B^*)$ to prevent queue buildup.

**Model-agnostic speedup.** Model inference scales readily via batching, but existing benchmarks run a single environment instance, making simulation the dominant bottleneck. Episode sharding closes this gap (Fig. 2): the model supply ceiling exceeds environment demand at all shard counts, so the speedup

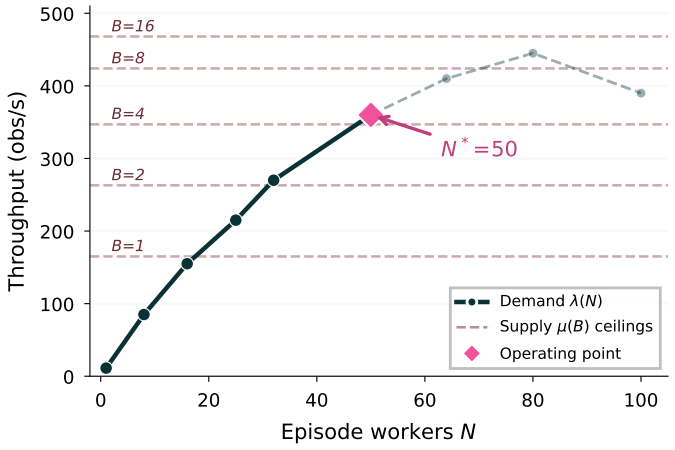


Fig. 2. Demand/supply throughput for LIBERO + CogACT [20] on H100. Dashed lines show supply ceilings $\mu(B)$ at each batch size. The operating point $K^* = 50$ uses 78% of the supply capacity at $B = 16$, leaving headroom to absorb burst arrivals and prevent queue buildup; beyond $K = 80$, environment overhead causes throughput to drop.

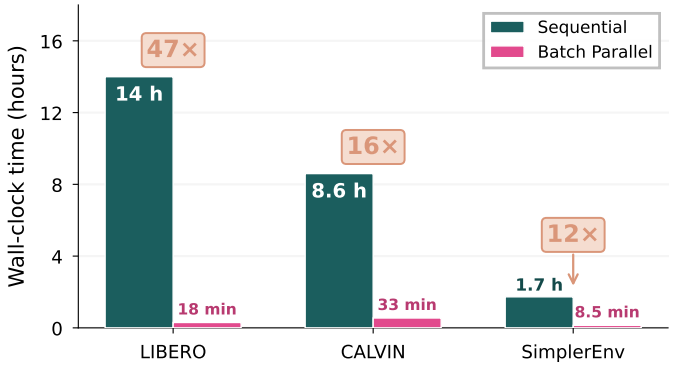


Fig. 3. Wall-clock evaluation time: sequential vs. batch parallel. LIBERO: 2,000 episodes, 50 shards, $B = 16$. CALVIN: 1,000 sequences, 16 shards. SimplerEnv: 288 episodes (3 seeds), 16 shards.

is determined by environment parallelism and transfers to any model.

On LIBERO with CogACT-7B (H100 model server, separate benchmark host), episode sharding from $K = 1$ to $K = 50$ increases environment throughput by 32.6× ($\lambda$: 11.2→364.6 observations per second (obs/s)), and batch inference from $B = 1$ to $B = 16$ increases model server throughput by 2.8× ($\mu$: 165.2→468.2 obs/s). Combined, 2,000 episodes complete in ∼18 minutes versus ∼14 hours sequentially, a **47× wall-clock speedup**. The same methodology applies to CALVIN (1,000 sequences, 16 shards, ∼33 min, 16× speedup) and SimplerEnv (288 episodes, 16 shards, ∼8.5 min, 12× speedup), as shown in Fig. 3.

## III. Validation

### A. Scope and Results

To validate the framework, we evaluate six published VLA codebases—OpenVLA [21], $\pi_{0.5}$ [2], OpenVLA-OFT [22], GR00T N1.6 [1], DB-CogACT [4], and X-VLA [3]—across three simulation benchmarks using fixed seeds and versioned Docker images from `ghcr.io`. Most VLA models require per-benchmark fine-tuning, so evaluation is only possible where public checkpoints exist. LIBERO: 4 suites × 10 tasks × 50 episodes (2,000 total). CALVIN: ABC→D, 1,000 chained sequences. SimplerEnv: 4 WidowX tasks, 24 episodes per task.

TABLE II
Reproduction matrix: ours (Δ vs. reported).

| Codebase | LIBERO (%) | CALVIN (len) | SimplerEnv (%) |
|---|---|---|---|
| OpenVLA | 76.2 (−0.3) | — | — |
| $\pi_{0.5}$ | 97.7 (+0.8) | — | — |
| OpenVLA-OFT | 96.7 (−0.4) | — | — |
| GR00T N1.6 | 94.9 (−2.1)† | — | 59.7 (−8.0)‡ |
| DB-CogACT | 94.7 (−0.2) | 4.02 (−0.04) | 63.5 (−6.0) |
| X-VLA | 97.4 (−0.7) | 4.30 (−0.13) | 94.8 (−1.0) |

— = no public checkpoint. †Community checkpoint. ‡Google Robot visual matching (others are WidowX).

Table II shows the reproduction matrix. Published scores largely reproduce across six codebases and three benchmarks, validating the framework's fidelity to reference implementations.

### B. Reproduction Challenges

These reproductions were non-trivial: single undocumented settings could cause catastrophic score changes.

Using the wrong proprioceptive state source in X-VLA [3] on LIBERO drops success rate from 97.8% to 42%, a 55 percentage point (pp) swing from one parameter. Confusing absolute and delta action modes (both valid 7D vectors, indistinguishable from data alone) produces 0% as positions accumulate and the robot diverges. OpenVLA-OFT [22] uses a quaternion-to-axis-angle conversion without antipodal normalization (angle $\in [0, 2\pi]$, matching robosuite convention), while our initial implementation flipped $w < 0$ quaternions (angle $\in [0, \pi]$); this single mismatch dropped LIBERO-Goal from 97% to 83% and LIBERO-Long from 95% to 56%. OpenVLA [21] applies a center crop (scale = 0.9) at evaluation time that is not documented in the paper; omitting it costs ∼3 pp. GR00T [1] expects end-effector pose as proprioceptive input, but this field exists only in an internal simulator fork, not in official SimplerEnv [9]; without it, scores drop from 30–55% to 0%.

Each of these was discovered only through systematic comparison of intermediate values against reference implementations. Reimplementing the necessary patches brought GR00T on SimplerEnv (Google Robot) from 0% to 59.7%, with a −8.0 pp gap to the reported score remaining.

## IV. VLA Leaderboard

Beyond framework validation, we compile a broader picture of the VLA evaluation landscape. We release a VLA leaderboard (Fig. 4) aggregating 657 results across 17 benchmarks and 509+ configurations, sourced from 1,704 papers that cite at least one of the tracked benchmarks.

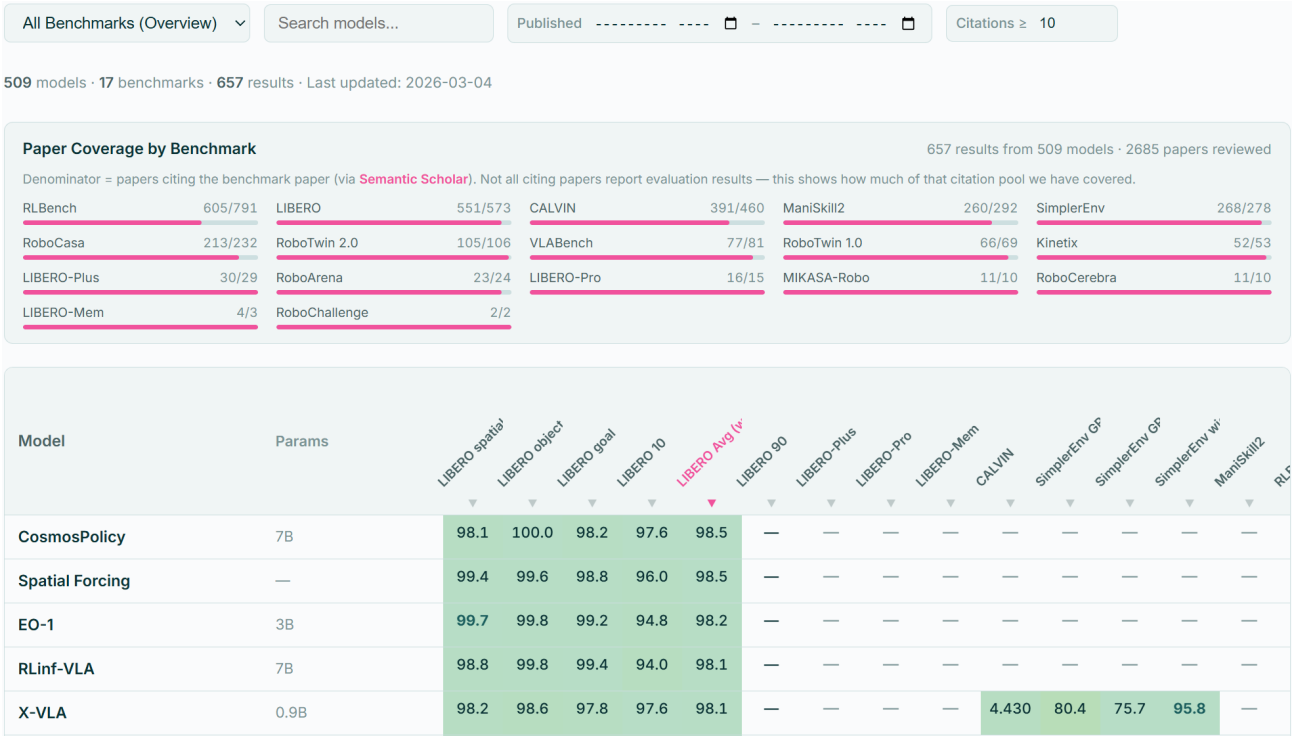


Fig. 4. VLA leaderboard (17 benchmarks, https://allenai.github.io/vla-evaluation-harness/leaderboard). Shown: models with >10 citations. Filterable by benchmark and model.

### A. Curation

Evaluation protocols vary across papers: SimplerEnv spans three incomparable robot configurations; CALVIN ABC→D and ABCD→D splits are not comparable; LIBERO papers report 4 or 5 suites. We established canonical protocol definitions for each benchmark, standardizing task subsets, metrics, splits, and comparability constraints.

An AI agent (Claude Code with Opus 4.6) reviewed 1,704 papers via MCP tool integrations (arXiv, Semantic Scholar, PDF reader) to extract and normalize results against these canonical protocols. A human operator then reviewed every entry, resolving anomalies and ambiguous cases. Each entry is versioned with full provenance metadata and validated against automated schema constraints, enabling fair cross-paper comparison.

### B. Cross-Benchmark Analysis

Fig. 5 shows the distribution of benchmark coverage across 509+ models and the 17 benchmarks tracked in the leaderboard: 81% are evaluated on a single benchmark, and only 6% on three or more. Cross-benchmark comparison is therefore rare, limiting our ability to assess general model capability across diverse environments and embodiments. This underscores the need for a unified framework that makes cross-benchmark comparison practical.

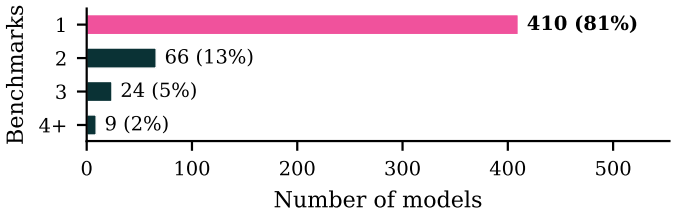


Fig. 5. Distribution of benchmark coverage per model. 81% of the 509+ models are evaluated on only one benchmark; only 3 (0.6%) on 5 or more.

## V. Conclusion

Validation across six codebases and three benchmarks confirms that `vla-eval` reproduces published scores within expected variance, while revealing that single undocumented parameters can shift results by tens of pp. `vla-eval` records the full evaluation configuration alongside every result, making any run reproducible from a single config file.

**Limitations.** Our audit covers six codebases across three simulation benchmarks; additional benchmarks and real-robot transfer are planned. Leaderboard results are extracted from published papers, not independently verified. Supported metrics are limited to task success rate; finer-grained dimensions such as subtask progress, task efficiency, and safety are not yet supported.

## References

[1] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan *et al.*, "GR00T N1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[2] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[3] J. Zheng, J. Li, Z. Wang, D. Liu, X. Kang, Y. Feng *et al.*, "X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model," *arXiv preprint arXiv:2510.10274*, 2025.

[4] B. Xie, E. Zhou, F. Jia, H. Shi, H. Fan, H. Zhang *et al.*, "Dexbotic: Open-source vision-language-action toolbox," *arXiv preprint arXiv:2510.23511*, 2025.

[5] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu *et al.*, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," in *NeurIPS Datasets and Benchmarks*, 2023.

[6] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu *et al.*, "ManiSkill2: A unified benchmark for generalizable manipulation skills," in *ICLR*, 2023.

[7] 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*, 2022.

[8] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi *et al.*, "The language model evaluation harness," 2024.

[9] X. Li, K. Hsu, J. Gu, O. Mees, K. Pertsch, H. R. Walke *et al.*, "Evaluating real-world robot manipulation policies in simulation," in *CoRL*, 2024.

[10] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020.

[11] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu *et al.*, "LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization," *arXiv preprint arXiv:2510.03827*, 2025.

[12] S. Han, B. Qiu, Y. Liao, S. Huang, C. Gao, S. Yan *et al.*, "RoboCerebra: A large-scale benchmark for long-horizon robotic manipulation evaluation," in *NeurIPS Datasets and Benchmarks*, 2025.

[13] M. Matthews, M. Beukman, C. Lu, and J. Foerster, "Kinetix: Investigating the training of general agents through open-ended physics-based control tasks," in *ICLR*, 2025.

[14] E. Cherepanov, N. Kachaev, A. K. Kovalev, and A. I. Panov, "Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning," *arXiv preprint arXiv:2502.10550*, 2025.

[15] N. Chung, T. Hanyu, T. Nguyen, H. Le, F. Bumgarner, D. M. H. Nguyen *et al.*, "Rethinking progression of memory state in robotic manipulation: An object-centric perspective," *arXiv preprint arXiv:2511.11478*, 2025.

[16] Y. Dai, H. Fu, J. Lee, Y. Liu, H. Zhang, J. Yang *et al.*, "RoboMME: Benchmarking and understanding memory for robotic generalist policies," *arXiv preprint arXiv:2603.04639*, 2026.

[17] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao *et al.*, "VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks," *arXiv preprint arXiv:2412.18194*, 2024.

[18] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Z. Li *et al.*, "RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation," *arXiv preprint arXiv:2506.18088*, 2025.

[19] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi *et al.*, "RoboCasa: Large-scale simulation of household tasks for generalist robots," in *RSS*, 2024.

[20] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao *et al.*, "CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation," *arXiv preprint arXiv:2411.19650*, 2024.

[21] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair *et al.*, "OpenVLA: An open-source vision-language-action model," in *CoRL*, 2024.

[22] M. J. Kim, C. Finn, and P. Liang, "Fine-tuning vision-language-action models: Optimizing speed and success," *arXiv preprint arXiv:2502.19645*, 2025.

[23] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn *et al.*, "$\pi_0$: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[24] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong *et al.*, "FAST: Efficient action tokenization for vision-language-action models," *arXiv preprint arXiv:2501.09747*, 2025.