LIBERO-LQ: A Latency- and Quality-Aware Benchmark for Vision–Language–Action Models

Tian Xie*
xie00529@umn.edu
University of Minnesota
Minneapolis, MN, USA

Youngjin Hong*
hong0745@umn.edu
University of Minnesota
Minneapolis, MN, USA

Zishen Wan
Georgia Institute of Technology
Atlanta, GA, USA

Yang (Katie) Zhao†
yangkatiezhao@umn.edu
University of Minnesota
Minneapolis, MN, USA

Abstract

Vision-Language-Action (VLA) models have shown strong performance on robotic manipulation tasks through large-scale multimodal pretraining. However, existing benchmarks typically assume negligible inference latency and rely primarily on binary success metrics, limiting their ability to evaluate deployment-relevant execution behavior. We introduce **LIBERO-LQ**, a **Latency- and Quality-aware** benchmark for evaluating VLA models under realistic execution constraints. LIBERO-LQ decouples policy inference from environment stepping, enabling simulated execution to progress during non-negligible inference latency and supporting systematic evaluation of synchronous and asynchronous execution paradigms under identical latency conditions. Beyond task success and completion time, LIBERO-LQ incorporates a suite of *trajectory-level, non-functional metrics* that capture execution quality, including smoothness, stability, efficiency, and energy consumption. Through experiments on representative VLA models, we show that latency-aware and quality-oriented evaluation exposes substantial differences in execution robustness and trajectory quality that are obscured under conventional synchronous evaluation.

Keywords: Vision-Language-Action Models, Embodied AI

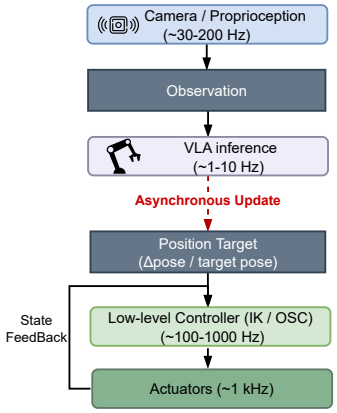

Figure 1. Execution pipeline of VLA under realistic deployment conditions. High-frequency sensing and low-level control operate continuously, while VLA policy inference runs at a much lower rate and updates position targets asynchronously. During inference latency, the robot continues executing previously issued commands via the low-level controller, resulting in decoupled policy computation and control execution. This mismatch is typically ignored in existing evaluation protocols.

1 Introduction

Vision-Language-Action (VLA) models have recently emerged as a powerful paradigm for general-purpose robotic manipulation. Representative systems such as RT-1 [5], RT-2 [4], OpenVLA [14], and π_0 [2] demonstrate impressive generalization by leveraging large-scale vision and language encoders. These models are commonly evaluated using standardized simulation benchmarks such as LIBERO [20], ManiSkill2 [9], BEHAVIOR-1K [15], and EmbodiedBench [28].

Existing evaluation protocols largely inherit assumptions from earlier imitation learning and reinforcement learning pipelines, in which policy inference is effectively instantaneous. While this assumption is reasonable for lightweight control policies, it no longer holds for modern VLA systems. In practice, VLA inference can take tens to hundreds of milliseconds, during which the robot continues executing previously issued commands or remains stationary. This decoupling between policy computation and control execution fundamentally alters the effective execution behavior, yet is not captured by current benchmark protocols.

Recent studies have shown that inference latency can substantially degrade VLA performance during deployment.

*Both authors contributed equally to this research.
†Corresponding author.

Delay-Aware Diffusion Policy [19] demonstrates a monotonic decrease in task success as execution latency increases due to observation-execution mismatch. AsyncVLA [12] further show that delayed and asynchronous execution necessitate real-time correction mechanisms to maintain performance. These results indicate that inference latency is a first-order factor affecting the real-world behavior of VLA policies. Beyond latency, binary task success provides only a coarse view of policy performance. Even when different VLA models achieve identical success rates, their underlying execution trajectories can vary substantially in terms of jerk and stability. Such differences are invisible to success-based evaluation yet critically affect deployability in real robotic systems. Finally, task success in embodied robotic manipulation is inherently closed-loop. Unlike other agent settings, robotic systems operate under strict real-time constraints, where excessive inference latency directly reduces effective control bandwidth. As a result, even a nominally correct policy can exhibit degraded closed-loop behavior when inference latency is ignored.

These observations raise a fundamental question: *Is it fair to compare VLA models using evaluations that ignore inference latency and execution-time behavior?* We argue that it is not. To address this gap, we propose a latency-aware benchmark for Vision-Language-Action models that explicitly decouples policy inference from control execution and models inference latency as a first-class experimental variable. By incorporating realistic execution semantics and evaluating policies beyond success rates alone, our benchmark provides a more faithful and fair assessment of VLA systems under realistic deployment conditions. **Our contributions are as follows**:

- We introduce **LIBERO-LQ**, a latency-aware benchmark for Vision-Language-Action (VLA) models that explicitly decouples policy inference from control execution to reflect realistic deployment conditions.
- We formalize latency-aware simulation and execution protocols that enable systematic evaluation of both synchronous and asynchronous VLA inference under diverse latency settings.
- Through extensive evaluation of representative VLA models, we show that LIBERO-LQ reveals substantial differences in execution quality that are obscured by conventional success-rate-based evaluation.

2 Background

Vision-Language-Action (VLA) models [4, 5, 14, 18] map visual observations and natural language instructions directly to robot actions using a learned policy. Recent systems have demonstrated strong task generalization by leveraging Vision-Language Model (VLM) backbones [1, 6, 16, 24]. By combining pretrained visual encoders [21] with large language models [25, 26] and training on large-scale robot demonstration datasets [7, 8, 13, 22, 27], VLA policies have emerged as generalist controllers capable of executing diverse manipulation tasks. As these models scale in size and capability, inference latency has become a central deployment challenge, fundamentally shaping how policy inference is coupled with action execution.

Importantly, VLA policies do not directly actuate robot motors, but instead generate high-level commands (e.g., end-effector targets or action chunks) that are executed by low-level controllers operating at much higher frequencies. As illustrated in Figure 1, perception and motor control loops typically run continuously, while VLA inference updates commands at a substantially lower rate. This temporal decoupling becomes particularly pronounced when inference latency is non-negligible.

Early VLA systems predominantly adopted *synchronous inference*, in which policy inference and execution are tightly coupled, causing blocking or stop-and-go behavior when inference latency is non-negligible [4, 5, 14]. To alleviate this limitation, more recent systems shift toward *asynchronous inference*, decoupling policy inference from control execution and allowing actions to be executed continuously while new predictions are generated in parallel. This paradigm, often implemented via frequent action chunking and receding-horizon control, has become common in modern VLA systems with substantial inference cost [2, 3, 11]. While these approaches improve deployability, they are typically evaluated under idealized or latency-agnostic simulation assumptions. Consequently, existing benchmarks do not systematically capture how inference latency and execution strategies interact with low-level control to affect trajectory-level behavior, motivating evaluation frameworks that explicitly model latency as a first-order system factor.

3 Libero-LQ

3.1 Latency-Aware Simulation

Conventional benchmarks freeze simulator time during policy inference and only advance simulation after inference completes, implicitly modeling inference latency as zero. In contrast, LIBERO-LQ adopts a latency-aware simulation design that explicitly decouples policy inference from environment stepping, allowing simulation to continue progressing during inference. Specifically, while policy inference incurs a measured wall-clock latency $\Delta t_{\text{inf}}^{\text{real}}$, the simulator advances through repeated `env.step` calls over an injected duration $\Delta t_{\text{inf}}^{\text{sim}}$. This design does not modify simulator dynamics or low-level controllers, but instead changes how high-level actions are scheduled during inference latency.

By explicitly modeling inference latency in simulation, our approach enables systematic analysis of trajectory quality, energy-related metrics, and robustness under realistic deployment constraints.

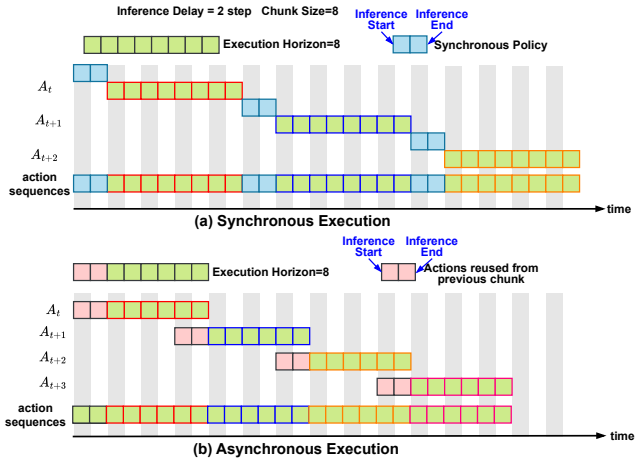

Figure 2. Comparison of synchronous and asynchronous policy execution under inference latency. **(a) Synchronous execution** blocks policy updates during inference, causing the robot to reuse the last available action chunk and exhibit stop-and-go behavior. **(b) Asynchronous execution** decouples inference from execution by continuously consuming actions from an execution buffer.

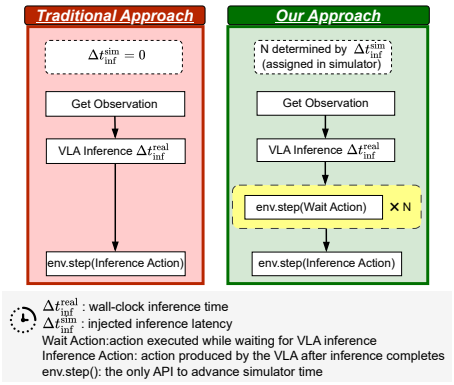

Figure 3. Left: Traditional evaluation tightly couples inference and environment stepping, implicitly assuming zero simulated inference latency ($\Delta t_{\text{inf}}^{\text{sim}} = 0$). **Right:** Our approach decouples inference from environment stepping by explicitly injecting simulated inference latency. During inference, the simulator advances via repeated `env.step` calls using a synchronous or asynchronous execution protocol, where the number of steps N is determined by $\Delta t_{\text{inf}}^{\text{sim}}$.

3.2 Policy Execution Protocol

Let Δt_c denote the control timestep and Δt_p the policy inference interval, with $\Delta t_p \gg \Delta t_c$. At each policy step k, the VLA policy produces either a single command or an action chunk

$$\mathcal{A}_k = \{a_k^{(1)}, a_k^{(2)}, \ldots, a_k^{(K)}\}, \tag{1}$$

which is executed by a low-level controller at the control frequency.

3.2.1 Synchronous Inference. In synchronous inference, policy inference blocks execution updates. During each inference interval, the controller tracks the most recent policy output:

$$u_t = \pi_{\text{ctrl}}(a_k, s_t), \quad t \in [k\Delta t_p, (k+1)\Delta t_p), \tag{2}$$

where a_k denotes the latest available policy output.

3.2.2 Asynchronous Inference. In asynchronous inference, policy inference and execution proceed in parallel. Execution consumes actions from a buffer $\mathcal{B}_t$ at the control frequency,

$$u_t = \mathcal{B}_t[0], \tag{3}$$

while policy inference asynchronously updates future actions in the buffer as new outputs become available. This formulation enables continuous execution under inference latency while preserving temporal misalignment effects arising from delayed observations. Please refer to Fig. 2 for details.

3.3 Metrics

Existing robotic manipulation benchmarks, including LIBERO, primarily evaluate policies using functional metrics such as binary task success. To address this limitation, we include several non-functional metrics, which characterize execution quality, efficiency, and robustness. As shown in Appendix A.

4 Experiments

4.1 Experimental Setup

We evaluate a diverse set of representative Vision-Language-Action (VLA) models that span different inference and execution protocols. Specifically, we consider:

- **OpenVLA**: An autoregressive VLA model inference at every decision step
- **OpenVLA-OFT**: An optimized variant of OpenVLA that employs autoregressive parallel decoding and action chunking.
- **π0.5**: State of the art Flow Matching-based VLA model, representing asynchronous inference with continuous execution and overlapping inference and control.
- **VLA-Cache**: An efficient VLA method. We include VLA-Cache as a representative efficient VLA approach to investigate whether efficiency-oriented optimizations affect trajectory quality.

Although our proposed **LIBERO-LQ** framework explicitly supports evaluation under non-zero inference latency, we additionally report results under **zero injected latency**, corresponding to the idealized assumption that policy inference completes instantaneously.

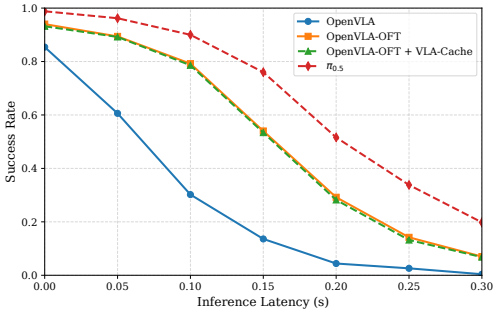

Figure 4. Success rate under injected inference latency.

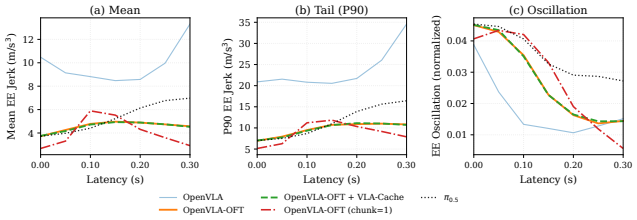

Figure 5. Trajectory quality under inference latency.

4.2 Experiment Results

4.2.1 Task Success Rate under Latency. As Figure 4 shows, We compare all VLAs under injected inference latency. OpenVLA degrades rapidly as inference latency increases, failing even under moderate latency. In contrast, OpenVLA-OFT is substantially more robust across the entire latency range. OpenVLA-OFT with VLA-Cache closely tracks the performance of OpenVLA-OFT, with only small differences across latency settings. The Flow Matching-based policy $\pi_{0.5}$ achieves the highest success rate throughout the latency sweep and exhibits the strongest robustness to increasing latency.

Moreover, the observed degradation in success rate and the differing rates of degradation across VLA models highlights that existing benchmarks implicitly assume instantaneous inference, leading to overly optimistic success-rate evaluations. Finally, OpenVLA-OFT with VLA-Cache consistently attains a slightly lower success rate than OpenVLA-OFT, and this performance gap remains almost identical as inference latency grows.

4.2.2 Trajectory Quality Analysis. Next, we analyze the relationship between trajectory smoothness and inference latency. Since OpenVLA performs policy inference at every control step, whereas OpenVLA-OFT employs action chunking, we introduce an additional OpenVLA-OFT variant with the action chunk size set to 1 to disentangle the effects of the OFT architecture itself from those induced by action chunking. We report both the mean jerk and the 90th-percentile (P90) jerk, as shown in Figure 5. OpenVLA exhibits substantially higher jerk than the other models, indicating less smooth motion, while simultaneously achieving the lowest end-effector oscillation. Moreover, introducing VLA-Cache does not lead to any noticeable degradation in trajectory smoothness. Finally, although $\pi_{0.5}$ shows slightly higher jerk than OpenVLA-OFT, its trajectory quality remains comparably smooth and well within acceptable ranges.

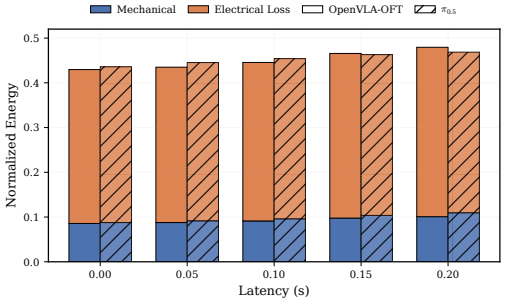

Figure 6. Normalized energy consumption decomposed into mechanical energy and electrical loss under increasing inference latency. For each latency, the left bar corresponds to OpenVLA-OFT and the right bar corresponds to the asynchronous reference policy $\pi_{0.5}$. Results show that both methods exhibit highly similar energy composition and per-step energy usage across latency settings.

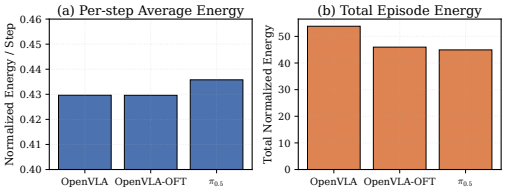

Figure 7. (a) Per-step normalized energy averaged over execution steps. (b) Total normalized energy accumulated over an episode. While OpenVLA exhibits per-step energy consumption comparable to OpenVLA-OFT and $\pi_{0.5}$, it incurs substantially higher total energy due to less efficient execution trajectories, highlighting the importance of evaluating total episode energy in addition to per-step metrics.

4.2.3 Energy Consumption Analysis. We next investigate the relationship between trajectory execution and energy consumption. For both π_0 and OpenVLA-OFT, we measure the energy expenditure associated with executing their trajectories. Since differences in success rates may lead to mismatched trajectories across methods, we report the energy normalized on a per-step basis to enable a fair comparison. As shown in Figure 6, these two baselines exhibit highly similar per-step energy consumption, indicating comparable energy usage at each execution step.

In addition, the estimated mechanical energy and electrical energy loss suggests that smoother, more conservative execution steps are often more energy-efficient on a per-step basis, but may result in increased total energy consumption due to longer execution horizon. To illustrate this effect, Figure 7 compares OpenVLA, OpenVLA-OFT, and $\pi_{0.5}$. While OpenVLA demonstrates lower per-step energy consumption, it incurs substantially higher total episode energy, highlighting that instead of instantaneous power usage, execution inefficiency dominates overall energy consumption.

References

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

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. 2024. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164* (2024). https://www.pi.website/blog/pi0

[3] Kevin Black, Manuel Y Galliker, and Sergey Levine. 2025. Real-Time Execution of Action Chunking Flow Policies. *arXiv preprint arXiv:2506.07339* (2025).

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. 2023. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *arXiv preprint arXiv:2307.15818*.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817* (2022).

[6] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. 2023. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565* (2023).

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

[8] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. 2023. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595* (2023).

[9] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. 2023. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659* (2023).

[10] Neville Hogan and Dagmar Sternad. 2009. Sensitivity of smoothness measures to movement duration, amplitude, and arrests. *Journal of motor behavior* 41, 6 (2009), 529–534.

[11] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. 2025. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. arXiv:2504.16054 [cs.LG] https://arxiv.org/abs/2504.16054

[12] Yuhua Jiang, Shuang Cheng, Yan Ding, Feifei Gao, and Biqing Qi. 2025. AsyncVLA: Asynchronous Flow Matching for Vision-Language-Action Models. *arXiv preprint arXiv:2511.14148* (2025).

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

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

[15] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. 2023. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*. PMLR, 80–93.

[16] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. 2024. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895* (2024).

[17] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. 2024. Evaluating Real-World Robot Manipulation Policies in Simulation. arXiv:2405.05941 [cs.RO] https://arxiv.org/abs/2405.05941

[18] Xiang Li, Cristina Mata, Jongwoo Park, Kumara Kahatapitiya, Yoo Sung Jang, Jinghuan Shang, Kanchana Ranasinghe, Ryan Burgert, Mu Cai, Yong Jae Lee, et al. 2024. Llara: Supercharging robot learning data for vision-language policy. *arXiv preprint arXiv:2406.20095* (2024).

[19] Aileen Liao, Dong-Ki Kim, Max Olan Smith, Ali-akbar Aghamohammadi, and Shayegan Omidshafiei. 2025. Delay-Aware Diffusion Policy: Bridging the Observation-Execution Gap in Dynamic Tasks. *arXiv preprint arXiv:2512.07697* (2025).

[20] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems* 36 (2023), 44776–44791.

[21] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193* (2023).

[22] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. 2024. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6892–6903.

[23] Valentyn Petrichenko, Lisa Lokstein, Gregor Thiele, and Kevin Haninger. 2024. Energy Consumption in Robotics: A Simplified Modeling Approach. arXiv:2411.03194 [cs.RO] https://arxiv.org/abs/2411.03194

[24] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, et al. 2024. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555* (2024).

[25] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118* (2024).

[26] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[27] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. 2023. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*. PMLR, 1723–1736.

[28] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. 2025. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560* (2025).

A Metrics

A.1 Inference Latency.

We distinguish between **real inference latency** $\Delta t_{\text{inf}}^{\text{real}}$, which measures the wall-clock time required for model inference on a given observation, and **simulated inference latency** $\Delta t_{\text{inf}}^{\text{sim}}$, which is an explicitly injected delay during simulation. The simulated latency is treated as an experimental variable and determines how long high-level actions are held while the simulator continues to advance.

A.2 Jerk.

To quantify trajectory smoothness, we measure *jerk*, defined as the third time derivative of the end-effector position, which captures abrupt changes in acceleration and is a standard indicator of motion smoothness in robotic manipulation. Let $\mathbf{p}_t \in \mathbb{R}^3$ denote the end-effector position at timestep t, sampled at a uniform interval Δt. We approximate the jerk vector using a third-order finite difference:

$$\mathbf{j}_t \approx \frac{\mathbf{p}_t - 3\mathbf{p}_{t-1} + 3\mathbf{p}_{t-2} - \mathbf{p}_{t-3}}{\Delta t^3}. \tag{4}$$

Trajectory smoothness is then quantified by the average jerk norm over the rollout:

$$J = \frac{1}{T-3} \sum_{t=4}^{T} \|\mathbf{j}_t\|_2, \tag{5}$$

where T denotes the trajectory length. Lower jerk norm values correspond to smoother and more stable motion, while higher values indicate abrupt or oscillatory behavior. [10]

A.3 Stability (End-Effector Oscillation).

While prior work [17] measures execution stability using jerk-based trajectory instability in joint space, we instead focus on execution-level oscillations in task space, which are particularly prominent under inference latency.

Specifically, we measure stability by the average deviation of the end-effector position from its mean over a short temporal window of the last K steps:

$$\bar{\mathbf{p}} = \frac{1}{K} \sum_{i=T-K+1}^{T} \mathbf{p}_i, \quad \text{EE-Osc} = \frac{1}{K} \sum_{i=T-K+1}^{T} \|\mathbf{p}_i - \bar{\mathbf{p}}\|_2$$

where $\mathbf{p}_i \in \mathbb{R}^3$ denotes the end-effector position at timestep i. Higher EE-Osc indicates increased oscillatory behavior during task execution, often arising from latency-induced control inconsistencies.

A.4 Trajectory Efficiency.

We assess trajectory efficiency by the total trajectory length required to successfully complete the task. Since the minimal feasible trajectory length depends on task geometry and goal configuration, efficiency is evaluated via relative, within-task comparisons among successful trials.

A.5 Energy Efficiency.

We evaluate energy efficiency using a trajectory-level energy metric computed from joint torques and velocities available in simulation. Specifically, the total trajectory energy E_{traj} is estimated by integrating mechanical power, electrical loss, and constant overhead terms over time using a simplified actuator energy model. To decouple energy consumption from execution length, we report normalized energy defined as $E_{\text{norm}} = E_{\text{traj}}/T$. Full details of the energy estimation model are provided in Appendix B.

B Energy Estimation Model

B.1 Trajectory-Level Energy Model

In simulation, joint torques and velocities are directly available, enabling trajectory-level energy estimation. Let $\boldsymbol{\tau}_t \in \mathbb{R}^J$ denote joint torques and $\dot{\boldsymbol{q}}_t \in \mathbb{R}^J$ joint velocities at timestep t. The instantaneous mechanical power is defined as

$$P_{\text{mech}}(t) = \boldsymbol{\tau}_t^\top \dot{\boldsymbol{q}}_t. \tag{6}$$

Since regenerative braking is not modeled, only positive mechanical power is considered:

$$P_{\text{mech}}^{+}(t) = \max\big(0,\ \boldsymbol{\tau}_t^\top \dot{\boldsymbol{q}}_t\big). \tag{7}$$

Electrical loss is approximated using a simplified DC motor model:

$$P_{\text{elec}}(t) = RK_t^{-2}\, \boldsymbol{\tau}_t^\top \boldsymbol{\tau}_t, \tag{8}$$

where RK_t^{-2} is a constant motor parameter. Following the pre-measured value in the previous research [23], we use $RK_t^{-2} = 0.0036 W/N^2 m^2$, and

A constant overhead power term P_{overhead} is added to account for controller and electronics. By measuring Franka

Emika Robot's idle power, we set $P_{\text{overhead}} = 85W$. The total trajectory-level energy is computed as

$$E_{\text{traj}} = \sum_{t=1}^{T} \left(P^{+}_{\text{mech}}(t) + P_{\text{elec}}(t) + P_{\text{overhead}} \right) \Delta t. \tag{9}$$

B.2 Normalized Energy Consumption

To decouple energy consumption from trajectory length, we additionally report normalized energy per execution step:

$$E_{\text{norm}} = \frac{E_{\text{traj}}}{T}. \tag{10}$$