Critical Interval MSE: Toward Reliable Offline Validation for Robot Manipulation Policies

Anonymous Author(s)
Affiliation
Address
email

Abstract: Real-world evaluation is the gold standard for robot manipulation policies because it tests them against the physical conditions and deployment challenges they are ultimately designed to handle. However, real-world evaluation is also the bottleneck for iterating on robot policies: it is costly, difficult to reproduce, and often too sparse to reliably compare nearby model variants. A straightforward proxy for performance is validation loss on expert demonstrations, but this proxy is often poorly correlated with real-world performance. In this paper, we introduce Critical Interval MSE (CI-MSE), an intuitively simple yet effective offline validation metric. CI-MSE restricts error computation to task-critical segments and pairs it with simple action-alignment procedures that better match rollout-time behavior. Across simulation and real-world experiments, CI-MSE yields a stronger correlation between validation error and rollout performance than raw MSE. Across a wide range of policy checkpoints, CI-MSE achieves a Spearman's rank correlation of -0.87, much closer to the ideal value of -1 than raw MSE's -0.61, demonstrating a significant improvement. We show through sensitivity analysis that our metric is robust to a wide range of hyperparameters. We further study the effectiveness of CI-MSE under evaluation distribution shifts and suggest design boundaries when using this metric. In summary, this paper provides a simple and reliable offline validation tool for accelerating policy iteration. Project webpage: https://ci-mse.github.io/

Keywords: offline validation, robot learning, benchmark design

1 Introduction

Rapid model iteration in robot learning depends on having a validation signal that is cheap, reproducible, and predictive of real-world behavior. In practice, the gold standard remains policy rollout on physical systems. However, such evaluation is expensive to run, hard to standardize across labs, and often limited to a small number of trials [1, 2, 3, 4]. These constraints make it difficult to compare nearby policy variants, especially when researchers are studying minor changes in architecture, dataset size, or training recipes [5, 6].

Offline validation is attractive because it can be computed on held-out demonstrations without additional rollout time on robots [7, 8]. Yet most practitioners agree that the common choice of action-space MSE often correlates weakly with actual task success [9, 8, 10, 11, 12, 13]. The mismatch arises partly because many timesteps in a trajectory are irrelevant to task completion but produce large action errors, while the truly consequential actions are brief, contact-rich, and sensitive to small errors [14, 15, 16]. Figure 1 illustrates this mismatch. Consider a robot gripper transferring a bottle. The grasp stage is closely related to task success, and the gripper must carefully align itself with the mouth of the bottle. In contrast, the transition stage is less relevant to task success: the gripper can move horizontally or lift the bottle higher before putting it down. However, compared with the expert trajectory, the validation error is disproportionately high for the transition stage because transition actions are diverse and take up a large portion of the trajectory. Therefore, averaged error is dominated by uncritical intervals, and performance signals are largely obscured. Moreover,

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

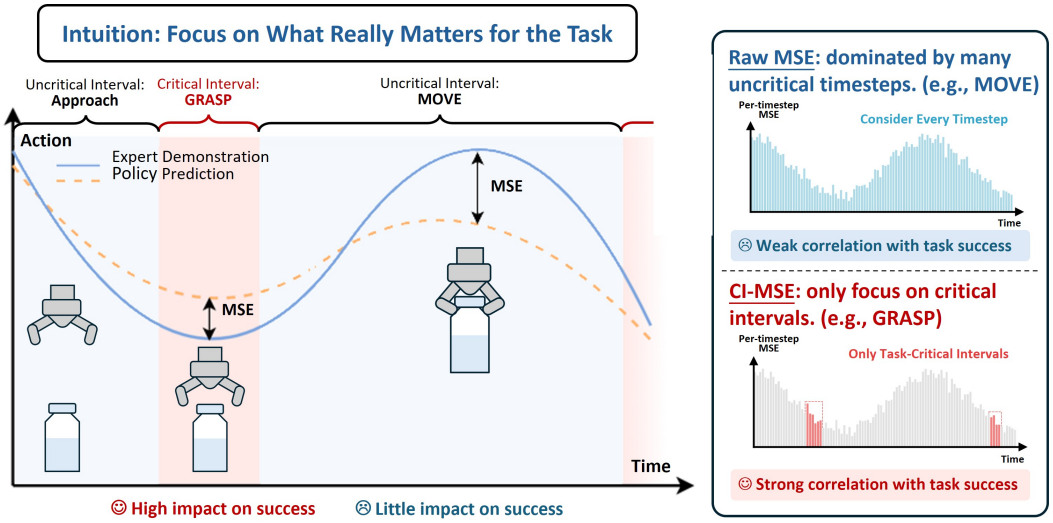


Figure 1: Illustration of critical intervals in a robot manipulation trajectory. This illustration shows an example of a robot gripper transferring a bottle. The aggregated MSE is dominated by uncritical move actions, while critical interval MSE focuses on the critical grasp stage.

rollout-time procedures such as temporal ensembling or real-time action chunking can materially alter policy behavior, even though they are usually ignored by offline metrics [17, 18].

To address this problem, this paper makes three contributions. First, we introduce *critical interval MSE* (CI-MSE), an offline validation metric that focuses on short task-critical intervals. Since uncritical actions dominate the validation error, an intuitively simple solution is to restrict error computation to task-critical intervals instead of averaging over entire demonstrations. We automatically identify task-critical intervals by leveraging a vision-language model for few-shot interval annotation. In addition, action alignment methods including temporal ensembling[17] / RTC[18] and dynamic time warping are applied to better match rollout-time behavior. Second, we assess the effectiveness of CI-MSE through both simulation and real-world experiments. We find that CI-MSE achieves stronger validation-evaluation correlation than raw MSE. Across 27 model checkpoints that differ in architecture, dataset size, training steps, and VLM backbone, CI-MSE achieves a Spearman's rank correlation of -0.87, while raw MSE only achieves -0.61. Real-world experiments on four tasks show results consistent with the simulation experiments. Third, we validate the robustness of CI-MSE through systematic experiments. We test our metric under different training-evaluation distribution shifts, a common setting when policy generalization is examined [19]. We find that although CI-MSE is affected by distribution shift, it still provides a more reliable ranking than raw MSE. We also show that our metric is robust to a wide range of hyperparameter selection through sensitivity analysis.

Our goal is not to claim that offline validation can replace gold standard robot rollouts. Rather, we aim to make offline validation more useful for swift model iteration, while clarifying the design principles for building more reliable offline benchmarks using our proposed metric.

2 Related Work

Real-world robot policy evaluation. Real-world benchmarks provide the most faithful signal for policy quality, and recent work improves reproducibility through standardized robot setups, autonomous evaluation, physical skill suites, and distributed pairwise comparisons [20, 1, 21, 3]. However, physical rollouts remain expensive, hard to replicate across labs, and difficult to distinguish many nearby model variants, especially under limited trial budgets.

Simulation-based evaluation. Simulation benchmarks offer reproducibility, parallelism, and controlled distribution shifts, enabling large-scale evaluation of manipulation policies [22, 23, 24, 19]. Notably, SIMPLER bridges the sim-to-real gap with careful parameter tuning and visual processing, and shows strong correlation with real-world performance across various policies [2]. However, extending simulation to new tasks and scenarios is time-consuming and requires domain expertise.

Offline evaluation and policy ranking. Compared with robot benchmark design, offline validation for robot policies is less established; related questions, however, have been studied extensively in RL and LLM evaluation. Offline policy evaluation in RL estimates policy value from fixed logged data, using estimators or benchmarks designed for settings where new interaction is costly [25, 26]. Since exact value prediction is often unnecessary for model selection, recent work also studies policy ranking directly [27, 28]. LLM evaluation has reached a related conclusion: scalable offline judges and pairwise comparisons are useful for ranking models, but must be checked against human-preference or task-level targets [29, 30].

3 Critical Interval MSE for Offline Validation

3.1 Problem Formulation

Let f denote a robot manipulation policy and let each validation trajectory be denoted as $\tau_i = \{(o_{i,t}, a_{i,t})\}_{t=1}^{T_i}$, where $o_{i,t}$ is the observation and $a_{i,t}$ is the expert action. The offline validation set is $\mathcal{D}_{\text{val}} = \{\tau_i\}_{i=1}^{N}$. Let $r(f)$ denote a rollout-based evaluation score for policy f, such as task success rate or a partial progress score. We seek an offline metric $L(f; \mathcal{D}_{\text{val}})$ whose induced ordering of policies agrees with the ordering under $r(f)$ as closely as possible. The metric L aggregates timestep-level errors $\{\ell(f, o_{i,t}, a_{i,t}) : 1 \le i \le N, 1 \le t \le T_i\}$ over the validation set.

$$\max_{L} corr\left(-L(f; \mathcal{D}_{\text{val}}), r(f)\right) \tag{1}$$

3.2 Critical Interval Filter

We define a *critical interval* as a contiguous segment of a demonstration in which action accuracy has a significant impact on task outcome. Typical examples include object contact, gripper grasping, insertion, or fine alignment near a task target. Actions in these intervals are under strict physical constraints and are therefore both sensitive to error and causally tied to success. By contrast, long transit motions or idle stabilization often contribute heavily to raw MSE without affecting whether the task succeeds. In practice, errors in uncritical intervals are $5 \sim 10$ times larger than errors in critical intervals. Consequently, aggregated error over the whole episode is dominated by uncritical intervals, and performance signals are largely obscured.

Given a set of critical intervals $\{\mathcal{I}_i\}_{i=1}^{N}$ on validation trajectories $\mathcal{D}_{\text{val}} = \{\tau_i\}_{i=1}^{N}$, CI-MSE computes action error only over timesteps inside those intervals:

$$\mathcal{D}_{\text{crit}} = \{(o_{i,t}, a_{i,t}) : \tau_i \in \mathcal{D}_{\text{val}}, t \in \mathcal{I}_i\} \tag{2}$$

In principle, we could give less weight to uncritical intervals and compute the weighted average error over the whole trajectory. However, tuning this weight complicates the metric and makes it much more fragile. Therefore, in this paper, we simply filter out uncritical timesteps. This straightforward change better aligns the metric with task structure by eliminating error contributions from uninformative timesteps.

We automate critical interval detection with few-shot VLM prompting. The first step is small-scale inspection of rollout failures, which provides intuition for the failure modes of the policy and what task phases are most consequential. The second step is automatic annotation with a vision-language model using few-shot prompting over demonstration videos. Appendix A.1 provides the template of the few-shot prompt. This produces a task-agnostic annotation pipeline that can be reused across datasets with modest human effort.

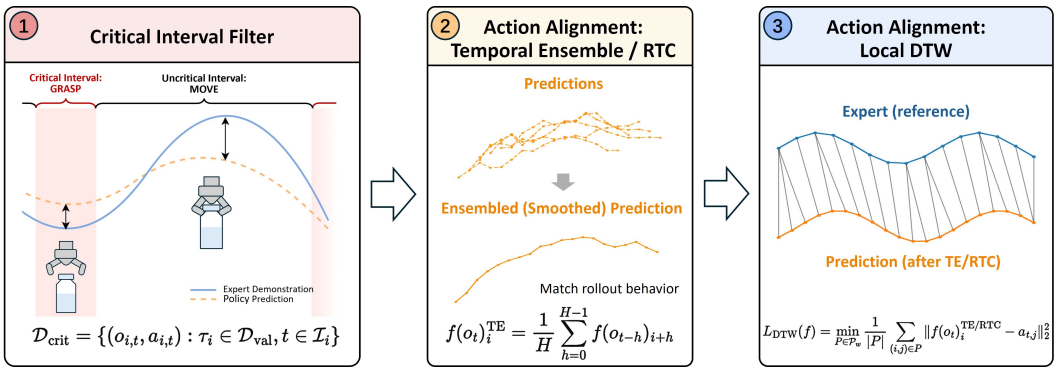


Figure 2: Pipeline for critical interval MSE computation. Critical intervals filter out uninformative timesteps. TE/RTC then matches rollout-time behavior. Finally, DTW computes distance to expert actions on a locally minimizing warping path.

3.3 Action Alignment for Offline Validation

Offline validation should reflect not only *what* actions a policy predicts, but also *how* those predictions are executed at test time. We therefore incorporate simple action-alignment procedures that better match rollout settings.

Temporal Ensembling and RTC. Inference-time methods such as temporal ensembling or real-time action chunking (RTC) are used during rollout to produce smooth execution. These methods can materially alter policy behavior. We therefore apply the same deployment-time inference methods to offline validation to better match rollout-time behavior and partially model the effect of compounding errors. To apply these procedures in offline validation, we validate over monotonic timesteps within a trajectory and apply temporal ensembling or RTC using overlapping action chunks from previous predictions. Temporal ensembling smooths rollout trajectories by averaging action predictions from overlapping chunks.

$$f(o_t)_i^{\text{TE}} = \frac{1}{H}\sum_{h=0}^{H-1} f(o_{t-h})_{i+h} \tag{3}$$

where $f(o_t)_i$ is the action for step i from the chunk predicted with observation o_t, and H is the ensemble horizon. Under high control frequency, temporal ensembling can also be interpreted as repeated sampling, which reduces variance in stochastic predictions. RTC guides the action generation process with overlapping action chunks. Let Δ be the number of delayed controller steps caused by inference latency during rollout. Applying RTC to offline validation can be viewed as inpainting $f(o_t)$ while softly matching the overlapping actions from $f(o_{t-\Delta})^{\text{RTC}}$. After the alignment with rollout-time TE/RTC, we compute validation error against expert actions on aligned action chunks.

Dynamic Time Warping (DTW). We use dynamic time warping (DTW) to compare predicted and expert action sequences under small temporal misalignments. Rather than matching each predicted action only to the expert action at the same timestep, DTW finds a monotone alignment path P that minimizes accumulated action error. For each critical timestep, we define the timestep-level CI-MSE as:

$$\ell_{\text{CI}}(f, o_t, a_t) = \min_{P\in\mathcal{P}_w} \frac{1}{|P|} \sum_{(p,q)\in P} \|f(o_t)_p^{\text{TE/RTC}} - a_{t,q}\|_2^2, \tag{4}$$

where $\mathcal{P}_w$ denotes the set of monotone warping paths satisfying $|p - q| \le w$ for window size w. This is useful because manipulation demonstrations often vary in tempo: two executions can make the same contact, grasp, or release slightly earlier or later while remaining behaviorally equivalent. A pointwise MSE would penalize such phase shifts even when the underlying action sequence is

Variant family	Values	Number of checkpoints
Architecture	$\pi_{0.5}$[32], X-VLA[33], Gr00t N1.7[34]	3
Data scale	20%, 40%, 60%, 80%, 100% of training data	5
Training steps	20k, 40k, 60k, 80k, 100k	5
PEFT	LoRA on {all, QKV, FFN} layers; rank {8, 16}	6
Action head	175M, 310M, 580M, 930M, 1220M parameters	5
VLM backbone	florence2-{base, large}, paligemma2-3B-{pt, mix}	4

Table 1: Simulation model variants. All non-architecture variants are based on X-VLA.

correct. We therefore apply DTW only as a local alignment correction, with w treated as a validation hyperparameter, so that the metric discounts harmless timing offsets without allowing large reorderings or failures to be hidden by excessive warping.

The dataset-level CI-MSE is then obtained by aggregating this timestep-level error over the critical validation set:

$$L_{\mathrm{CI}}(f; \mathcal{D}_{\mathrm{val}}) = \mathcal{A}\left(\{\ell_{\mathrm{CI}}(f, o_{i,t}, a_{i,t}) : (o_{i,t}, a_{i,t}) \in \mathcal{D}_{\mathrm{crit}}\}\right). \tag{5}$$

Here $\mathcal{A}$ denotes an aggregation operator. In our experiments, we use median aggregation for both CI-MSE and raw MSE.

3.4 Stable Rollout Ranking with Elo

In real-world experiments, researchers often use partial success or progress scores when binary success rates are too coarse and available trial counts are small. These scores return signals even if the task is not completed and serve as nuanced signals of policy quality. However, such scalar scores implicitly assume a linear relationship between annotated progress and true performance, which may be unjustified. We therefore recommend ranking policies through pairwise comparisons across trials. We introduce an Elo-style ranking system that rates policies by updating their scores after each trial based on the expected outcome versus the actual result. Without assuming numerical gaps of trial outcomes, this produces a more stable leaderboard under limited rollout budgets.

4 Experiments

4.1 Experimental Setup

We evaluate CI-MSE in both simulation and real-world settings. In simulation, we use the LBM-Eval benchmark and associated datasets [31], which consist of 49 tasks and $\sim$10k demonstrations. We compare controlled vision-language-action policy variants that differ in architecture, dataset scale, training steps, parameter-efficient finetuning, action-head size, and VLM backbone, as summarized in Table 1. In the real world, we study diffusion policies trained on data-scaling datasets [9] and evaluated on a Franka arm across pour-water, arrange-mouse, fold-towel, and unplug tasks. Table 4 lists the CI-MSE hyperparameters for both settings. The only difference between the simulation and real-world hyperparameters is the evaluated chunk steps, because of different controller frequency and predicted action chunk lengths. The training and validation set compositions are described in Appendix A.3.

For simulation, rollout performance is measured by success rate. For real-world experiments, where policy gaps are smaller and trial budgets are tighter, we use Elo scores as the main rollout score and compare them against partial success scores; Appendix A.2 analyzes the stability of these rollout scores. Offline validation compares raw MSE with CI-MSE.

4.2 Correlation between Validation Error and Rollout Performance in Simulation

Our first experiment measures how well offline metrics predict rollout outcomes across VLA model variants. Overall, Figure 3 shows a substantial improvement in the validation-rollout agreement. Across 27 policies, CI-MSE achieves a Pearson correlation of $r = -0.74$ and a Spearman correlation of $\rho = -0.87$, while raw MSE only reaches $r = -0.56$ and $\rho = -0.61$. Table 2 gives the

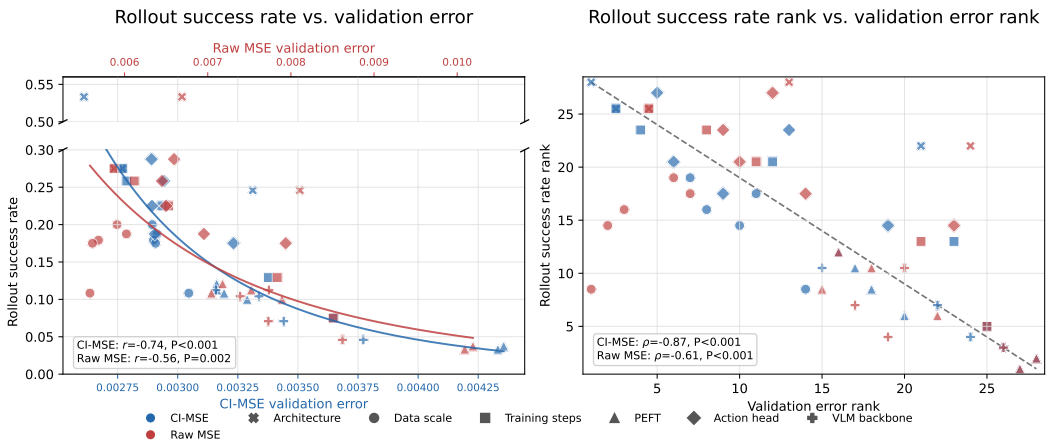


Figure 3: Simulation evaluation-validation correlation. Left: evaluation success rate versus validation error, with CI-MSE and raw MSE shown on aligned validation-error scales. The fitting curve follows the power law $y = ax^b$. Right: rank correlation between success-rate rank and validation-error rank. Marker color indicates the validation metric, and marker shape indicates the model-variant group.

Validation metric	Model variants											
	Architecture		Data scale		Training steps		PEFT		Action head		VLM backbone	
	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$
Critical Interval MSE	**-0.74**	**-1.00**	**-0.97**	**-0.70**	-1.00	-1.00	**-0.99**	**-0.94**	-0.60	-0.70	**-0.96**	**-1.00**
Raw MSE	-0.24	-0.50	0.67	0.90	-1.00	-1.00	-0.99	-0.77	**-0.75**	-0.70	-0.83	-0.40

Table 2: Correlation between validation error and rollout success rate across model variants in simulation. We report both Pearson's r and Spearman's ρ correlation coefficients. Blue cells indicate negative correlation, and orange cells indicate positive correlation. Color intensity indicates the magnitude of the correlation. Best values are closest to -1, and are highlighted in bold.

corresponding Pearson and Spearman correlations by variant family, showing that the validation-evaluation correlation is not uniform across variant types. For example, when varying training steps, both metrics are nearly perfectly rank-consistent with rollout success. However, when varying data scale, raw MSE is inverted and has a positive correlation coefficient ($r = 0.67$, $\rho = 0.90$), even though lower validation error should predict higher rollout success. On the other hand, CI-MSE achieves a Spearman correlation at least as strong as raw MSE's across all variant families. For the data scale variant, where raw MSE gives the wrong ordering, CI-MSE achieves a Pearson correlation of $r = -0.98$ and a Spearman correlation of $\rho = -0.90$, which is a significantly stronger signal.

This suggests that raw MSE's reliability is limited to certain variant families. When model quality changes through data scale, backbone choice, or other factors that alter policy behavior, averaging error over all timesteps irrelevant motion and obscure success signal. CI-MSE alleviates this issue and therefore is applicable to all examined variant families.

4.3 Distribution Shift in Policy Evaluation

We next study a harder regime in which validation and evaluation are matched but differ from the training distribution. This setting captures a common use case: researchers want to validate generalization out-of-distribution (OOD). Table 3 is structured around three distribution-shift dimensions: object layout OOD, visual OOD, and skill OOD. For object layout OOD, the objects are spawned outside of the bounding box of the training dataset. For visual OOD, we change the background and table texture. For skill OOD, we select 5 tasks unseen in the training set.

Validation metric	Collectors matched								Collectors mismatched							
	Pour water				Arrange mouse				Fold towel				Unplug			
	Env.		Obj.		Env.		Obj.		Env.		Obj.		Env.		Obj.	
	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$
Critical Interval MSE	**-0.99**	**-1.00**	**-0.99**	**-1.00**	**-0.96**	**-1.00**	**-0.66**	**-0.40**	**-0.05**	**-0.40**	**-0.87**	**-1.00**	**0.48**	0.40	-0.53	-0.60
Raw MSE	-0.47	-0.80	-0.86	-1.00	-0.80	-0.80	-0.06	-0.20	0.27	0.60	-0.09	0.00	0.62	0.40	**-0.86**	**-0.80**

Table 5: Correlation between validation error and real-world rollout Elo score. Each task is evaluated across unseen environments (Env.) and unseen objects (Obj.).

Table 3 shows that CI-MSE remains more predictive than raw MSE across the object layout and skill OOD settings. The advantage is largest for skill shift, where CI-MSE improves rank correlation from $\rho = -0.36$ to $\rho = -0.69$. The correlations for skill and visual OOD are relatively low because many model variants achieve near-zero success rates, making these variants hard to separate. Object-layout shift gives the strongest overall agreement for both metrics, but CI-MSE still improves Spearman correlation from $\rho = -0.77$ to $\rho = -0.88$. Visual shift is the hardest case to separate: both metrics degrade, and the margin between CI-MSE and raw MSE is smaller. Researchers should use CI-MSE cautiously when evaluating visual and skill-level generalization.

Validation metric	Distribution shift types					
	Object layout		Visual		Skill	
	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$	$r \downarrow$	$\rho \downarrow$
Critical Interval MSE	**-0.81**	**-0.88**	**-0.62**	-0.66	**-0.48**	**-0.69**
Raw MSE	-0.77	-0.77	-0.58	**-0.68**	-0.29	-0.36

Table 3: Correlation between validation error and rollout success rate under evaluation distribution shift.

Hyperparameters	Simulation	Real-world
Ensemble horizon	8	8
DTW window size	1	1
Evaluated chunk steps	0–24	2–8

Table 4: CI-MSE hyperparameters. Evaluated chunk steps denote the action indices in a predicted chunk used for execution or validation error computation.

4.4 Validation-Rollout Correlation and Cross-Domain Effects in Real-World Experiments

The real-world experiments test whether the simulation trend carries over to real physical evaluation and limited trials. Evaluated diffusion policies are trained on 2^m randomly selected object-environment pairs ($m = 2, 3, 4, 5$) [9]. Table 5 reports the corresponding correlations. CI-MSE is the stronger predictor for most cases: it achieves near-perfect cross-environment agreement for pour water ($r = -0.99$, $\rho = -1.00$) and arrange mouse ($r = -0.96$, $\rho = -1.00$), and it substantially improves fold-towel cross-object validation ($r = -0.87$, $\rho = -1.00$) over raw MSE ($r = -0.09$, $\rho = 0.00$).

We also investigate the effect of data collectors' action styles on validation error. The fold-towel and unplug settings introduce an additional practical confounder: because the original dataset does not include validation sets for these two tasks, their validation datasets were collected by operators who were different from those who collected the training datasets. This makes the validation error harder to interpret: a policy can receive a higher offline error because it does not match the validation operator's style, even if its rollout behavior remains competitive under the evaluation protocol. Under this confounder, correlations are weaker and less consistent. This indicates that offline validation is sensitive to collectors' action modes, and consistent data-collection protocols are important when using action-space validation error as a proxy for real-world performance.

4.5 Sensitivity Analysis

We perform a sensitivity analysis of CI-MSE to its hyperparameters and critical interval annotation. We vary the ensemble horizon, evaluated action chunk steps, DTW window size, and critical interval length, and we shift the critical interval. This sweep also includes component ablations: setting the

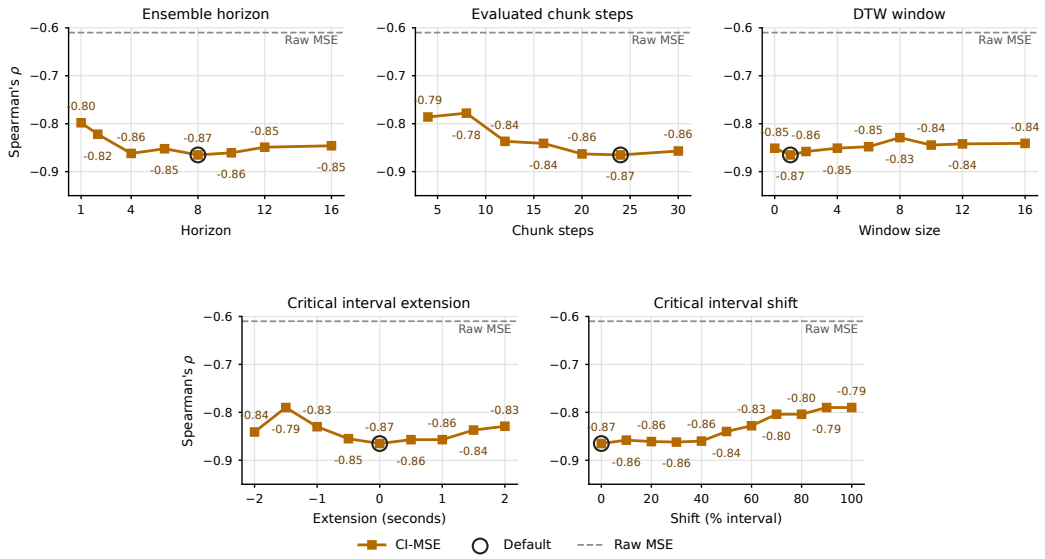


Figure 4: Sensitivity analysis of CI-MSE to its hyperparameters and interval annotation.

ensemble horizon to $H = 1$ removes temporal ensembling, and setting the DTW window size to $W = 0$ removes DTW. We then compute the correlation between CI-MSE and rollout success rate using the same model checkpoints and validation set as Section 4.2. Figure 4 shows that CI-MSE is relatively insensitive to these choices across a reasonable range of hyperparameters and consistently outperforms raw MSE.

The sensitivity trends are consistent with the design of CI-MSE. Temporal ensembling improves correlation from the no-ensembling ablation ($H = 1$, $\rho = -0.80$) to $H = 8$ ($\rho = -0.87$), after which the curve remains strong but slightly weaker, suggesting that moderate smoothing better matches rollout-time execution without over-smoothing the action sequence. DTW window size $W = 1$ gives the best result compared with the no-warping ablation at $W = 0$, while larger windows gradually weaken the correlation. For interval length, slightly extending or shortening the critical interval barely degrades the correlation. Shifting the critical interval shows a similar pattern: correlation remains stable when the shift magnitude is less than 40% of the interval length and starts to degrade when the shift magnitude is larger than 40%. This indicates that the critical interval is the core of our metric and that it tolerates moderate annotation imprecision. Overall, the metric is not brittle in the useful operating range, indicating that the improvement is not driven by a narrowly tuned hyperparameter choice.

5 Conclusion and Limitations

This paper introduced *critical interval MSE*, an offline validation metric that filters task-irrelevant action errors and matches rollout-time behavior. Across simulation and real-world experiments, CI-MSE correlates better with rollout outcomes than raw MSE, including under several distribution shifts, and remains stable across reasonable hyperparameter choices.

Limitations. CI-MSE still inherits the limits of offline validation: it does not observe dynamics, so it can misread policies that solve a task through valid approaches not represented in the demonstrations. It assumes a reasonably consistent action mode between training and validation; operator or collection-protocol mismatch can affect validation quality. In addition, our current focus is on short-horizon manipulation; CI-MSE is less suitable for tasks that depend on long-horizon planning. Despite these limitations, our method is applicable to a wide range of manipulation tasks and offers a practical tool for comparing nearby model variants and accelerating policy iteration.

References

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

[2] 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 8th Conference on Robot Learning*, pages 3705–3728, 2025.

[3] P. Atreya, K. Pertsch, T. Lee, M. J. Kim, A. Jain, A. Kuramshin, C. Eppner, C. Neary, E. Hu, F. Ramos, J. Tremblay, K. Arora, K. Ellis, L. Macesanu, M. Leonard, M. Cho, O. Aslan, S. Dass, J. Wang, X. Yuan, X. Yang, A. Gupta, D. Jayaraman, G. Berseth, K. Daniilidis, R. Martin-Martin, Y. Lee, P. Liang, C. Finn, and S. Levine. Roboarena: Distributed real-world evaluation of generalist robot policies. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 336–364. PMLR, 2025. URL https://proceedings.mlr.press/v305/atreya25a.html.

[4] Y. Li, Y. Zhu, J. Wen, C. Shen, and Y. Xu. Worldeval: World model as real-world robot policies evaluator. *arXiv preprint arXiv:2505.19017*, 2025. URL https://arxiv.org/abs/2505.19017.

[5] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024. doi:10.15607/RSS.2024.XX.120. URL https://arxiv.org/abs/2403.12945.

[6] Open X-Embodiment Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. URL https://arxiv.org/abs/2310.08864.

[7] L. Hussenot, M. Andrychowicz, D. Vincent, R. Dadashi, A. Raichuk, S. Ramos, N. Momchev, S. Girgin, R. Marinier, L. Stafiniak, M. Orsini, O. Bachem, M. Geist, and O. Pietquin. Hyperparameter selection for imitation learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4511–4522. PMLR, 2021. URL https://proceedings.mlr.press/v139/hussenot21a.html.

[8] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

[9] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pages 54877–54910, 2025.

[10] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 158–168. PMLR, 2022. URL https://proceedings.mlr.press/v164/florence22a.html.

[11] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2022. URL https://arxiv.org/abs/2112.01511.

[12] A. Bronars, Y. Park, and P. Agrawal. Tune to learn: How controller gains shape robot policy learning. *arXiv preprint arXiv:2604.02523*, 2026. URL https://arxiv.org/abs/2604.02523.

[13] M. Tiezzi, T. Apicella, C. Cardenas-Perez, G. Fregonese, S. Dafarra, P. Morerio, D. Pucci, and A. Del Bue. Learning to evaluate autonomous behaviour in human-robot interaction. *arXiv preprint arXiv:2507.06404*, 2025. URL https://arxiv.org/abs/2507.06404.

[14] C. Wen, J. Lin, J. Qian, Y. Gao, and D. Jayaraman. Keyframe-focused visual imitation learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11123–11133. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wen21d.html.

[15] E. Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4613–4619, 2021. doi:10.1109/ICRA48506.2021.9560942. URL https://arxiv.org/abs/2105.06411.

[16] T. Tsuji, Y. Kato, G. Solak, H. Zhang, T. Petrič, F. Nori, and A. Ajoudani. A survey on imitation learning for contact-rich tasks in robotics. *The International Journal of Robotics Research*, 2026. doi:10.1177/02783649261417694. URL https://doi.org/10.1177/02783649261417694.

[17] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023. doi:10.15607/RSS.2023.XIX.016. URL https://arxiv.org/abs/2304.13705.

[18] K. Black, M. Galliker, and S. Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

[19] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024. URL https://arxiv.org/abs/2402.08191.

[20] G. Zhou, V. Dean, M. K. Srirama, A. Rajeswaran, J. Pari, K. Hatch, A. Jain, T. Yu, P. Abbeel, L. Pinto, C. Finn, and A. Gupta. Train offline, test online: A real robot learning benchmark. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9197–9203, 2023. doi:10.1109/ICRA48891.2023.10160594.

[21] Y. Chen, K. Kimble, E. H. Adelson, T. Asfour, P. Chanrungmaneekul, S. Chitta, Y. Chitambar, Z. Chen, K. Goldberg, D. Kragic, H. Li, X. Li, Y. Li, A. Prather, N. Pollard, M. A. Roa-Garzon, R. Seney, S. Sha, S. Wang, Y. Xiang, K. Zhang, Y. Zhu, and K. Hang. Manipulationnet: An infrastructure for benchmarking real-world robot manipulation with physical skill challenges and embodied multimodal reasoning. *arXiv preprint arXiv:2603.04363*, 2026.

[22] 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. doi:10.1109/LRA.2020.2974707.

[23] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

[24] 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*, 2023.

[25] N. Jiang and L. Li. Doubly robust off-policy value evaluation for reinforcement learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 652–661, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/jiang16.html.

[26] J. Fu, M. Norouzi, O. Nachum, G. Tucker, Z. Wang, A. Novikov, M. Yang, M. R. Zhang, Y. Chen, A. Kumar, C. Paduraru, S. Levine, and T. Le Paine. Benchmarks for deep off-policy evaluation. In *International Conference on Learning Representations*, 2021.

[27] L. Da, P. Jenkins, T. Schwantes, J. Dotson, and H. Wei. Probabilistic offline policy ranking with approximate bayesian computation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20370–20378, 2024.

[28] P. Gu, M. Zhao, X. He, Y. Cai, and B. An. Porank: A practical framework for learning to rank policies. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 4044–4052, 2024. doi:10.24963/ijcai.2024/447.

[29] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023. Datasets and Benchmarks Track.

[30] S. Tan, S. Zhuang, K. Montgomery, W. Tang, A. Cuadron, C. Wang, R. Popa, and I. Stoica. Judgebench: A benchmark for evaluating llm-based judges. In *International Conference on Learning Representations*, 2025.

[31] J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026.

[32] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

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

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

A Appendix

A.1 Few-Shot Prompt for Critical Interval Annotation

We use the following prompt template to ask the vision-language model to annotate task-critical intervals from demonstration videos. The template specifies the semantic definitions of each interval, constrains timestamps to one decimal place, and requires a JSON-only response. Highlighted text marks variable fields that are changed for each annotation task. Gemini 2.5 Pro is used for annotation.

```
You are given a short robot manipulation video.
Your task is to annotate 2 specific time intervals in the video with timestamps in seconds, accurate to
     one decimal place.

The 2 intervals are defined semantically as:

Interval 1
    - Start: The gripper has not yet grabbed the cup, and its tip is about 5 cm away from the cup.
    - End: The gripper closes and has just fully gripped the cup, slightly lifting it up.
Interval 2
    - Start: The gripper is holding the cup and about 10 cm away from the coaster.
    - End: The gripper releases the cup so that it now rests by the coaster.

Requirements:
- Return exactly 2 intervals in a JSON object.
- Each interval must have:
  - "label": a short description.
  - "start": start time in seconds, with exactly one decimal place (e.g., 2.3).
  - "end": end time in seconds, with exactly one decimal place.
- Timestamps must be within the video duration.
- If you are uncertain, make your best estimate based on the visual evidence, but still output valid
     timestamps.
- If there are retries, only return the last attempt's timestamps.

Output format:
Return ONLY a valid JSON object, with no extra text, no explanations, and no Markdown code fences.

The JSON format must be exactly:

{
  "intervals": [
    {
      "label": "<description text>",
      "start": <time in seconds, 1 decimal place>,
      "end": <time in seconds, 1 decimal place>
    },
    ... one object per interval above ...
  ]
}

Here is an example of how to label the intervals and the corresponding video in the attachment.
<example video>
Example response:

{
  "intervals": [
    {
      "label": "The gripper grabs the cup from the table.",
      "start": 4.1,
      "end": 6.3
    },
    {
      "label": "The gripper places the cup by the coaster.",
      "start": 6.9,
      "end": 8.2
    }
  ]
}

Please label the critical intervals in this video using the previous examples.
<video>
```

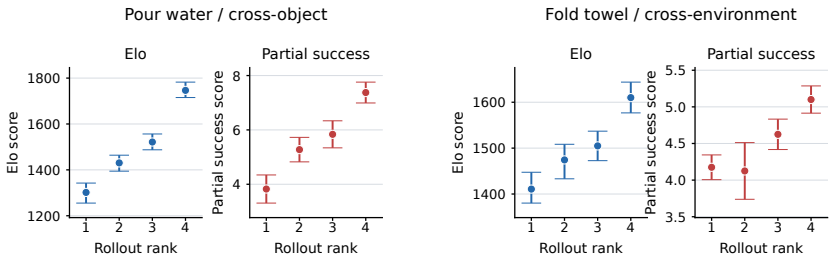


Figure 5: Confidence intervals for Elo scores and partial success scores. The confidence intervals are computed via 1,000 bootstrap resamples. Error bars denote 95% confidence intervals. Models are ordered by rollout rank.

A.2 Revisiting Rollout Scores

We examine the stability of the real-world rollout metric itself. We estimate confidence intervals for Elo scores and partial success scores using 1,000 bootstrap resamples. As shown in Figure 5, Elo produces more stable model orderings than partial success scores in the pour-water cross-object and fold-towel cross-environment settings. Pairwise Elo comparisons reduce dependence on arbitrary progress-score scales and provide a more stable target for correlation analysis.

A.3 Composition of Training and Validation Sets

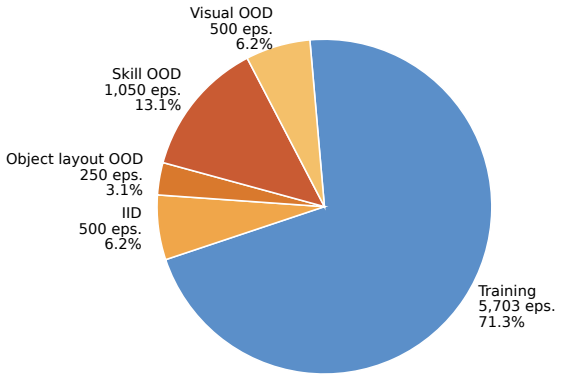


Figure 6: Episode-level composition of the LBM-Eval dataset used in the simulation experiments. The dataset contains 8,003 episodes in total; labels report both episode count and percentage.

For simulation experiments, we use the LBM-Eval dataset [31], which contains 49 tasks and $\sim$10k high-quality demonstrations collected through teleoperation in simulation. We split the dataset into a training set and validation sets with different distribution shifts. Figure 6 shows the composition of the training and validation sets. Figure 7 shows the process of partitioning the dataset into a training set and validation sets. We first exclude 5 skills as the skill OOD validation set. The objects in the original dataset are spawned uniformly in a rectangular bounding box. We create a smaller bounding box inside the original bounding box, use episodes with objects spawned outside the smaller box as the object layout OOD validation set, and split the remaining episodes into the training set and the IID validation set. Finally, we swap the background and table texture into 15×6 new combinations, as shown in Figure 8, and replay the episodes in the simulator to obtain the visual OOD validation set.

For real-world experiments, we use the data-scaling datasets from [9], which contain 4 tasks and 32 object-environment pairs for each task, collected using UMI. For the pour-water and arrange-mouse

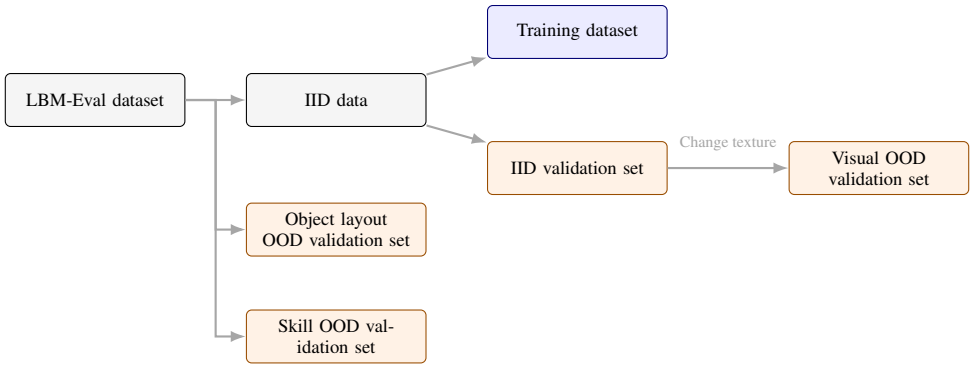


Figure 7: Composition of the LBM-Eval training and validation sets used in the simulation experiments.

tasks, validation sets are provided by the authors. The cross-object validation sets contain 8 objects in the same environment. The cross-environment validation sets contain 8 environments with the same object. We use 30 episodes per object/environment for validation. For the fold-towel and unplug tasks, validation sets are not provided, so we collected an equal amount of validation data ourselves. As a result, the collectors differ between the training and validation sets, inducing an extra confounder that affects the validation error.

A.4 Details of Rollout Evaluation Settings

A.4.1 Simulation Settings

For simulation experiments, we use the simulator and scenarios from the LBM-Eval suite. We test the policies' success rates across 12 seen tasks and 5 unseen tasks, following the validation-set splits. When evaluating under distribution shift, careful control of scenario generation is required to ensure that the distribution shift matches the validation set and differs from the training set, especially for the object layout and visual OOD settings. For object layout OOD, we use rejection sampling to generate object layouts. We sample inside the outer bounding box and reject samples in which objects are spawned inside the inner bounding box. For visual OOD, we prepare unseen environment maps and table textures, as shown in Figure 8, and then swap the background and table texture into 15×6 new combinations.

The main metric for measuring rollout performance is success rate. We evaluate each policy for 20 trials per skill and report the average success rate across all skills.

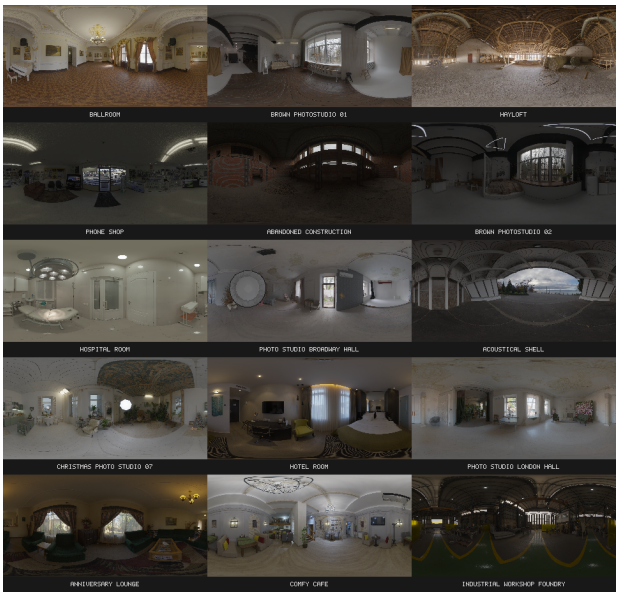
A.4.2 Real-World Settings

The real-world evaluation is conducted on a Franka arm across two categories: 8 objects and 8 environments. The trial budget is 10 trials per model per object/environment. We arrange objects according to a fixed grid layout so that different models face consistent initial states, ensuring fair pairwise comparisons.

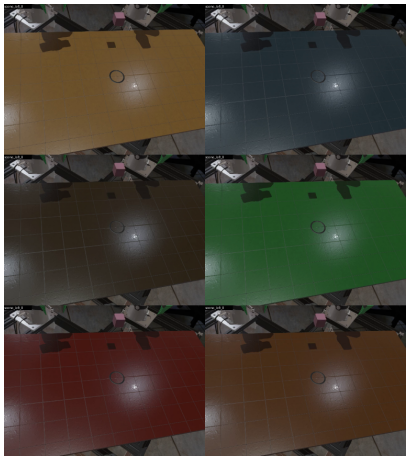
For real-world rollout evaluation, we introduce Elo ranking to obtain more stable rollout scores under limited trial budgets and noisy success rates. We convert each trial into an ordinal outcome level for Elo comparison. These levels define an ordering only: in a pairwise comparison, the policy with the higher outcome level is treated as the winner, while equal levels are treated as ties. We do not interpret the numerical gaps between adjacent levels as equal distances, and we do not use these levels as scalar partial-success scores.

Pour Water.

Step 1: Grasping the drink bottle.



(a) Environment maps



(b) Table textures

Figure 8: Environment and table textures used in visual distribution shift evaluation.

- **Level 0**: The gripper does not approach the drink bottle.
- **Level 1**: The gripper touches the drink bottle but does not grasp it due to minor errors, or it initially grasps the bottle but the bottle slips out during lifting.
- **Level 2**: The gripper pushes the drink bottle a significant distance before grasping it, or the bottle slips during the following process.
- **Level 3**: The gripper successfully grasps the drink bottle without any slippage.

Step 2: Pouring water into the mug.

- **Level 0**: The gripper does not approach the mug.
- **Level 1**: After rotating the drink bottle, its mouth remains outside the mug, making pouring impossible, or the robot knocks over the mug.
- **Level 2**: After rotating the drink bottle, its mouth is positioned just above the rim of the mug, allowing only partial pouring.
- **Level 3**: After rotating the drink bottle, its mouth is completely inside the mug, facilitating complete pouring.

Step 3: Placing the bottle on the red coaster.

- **Level 0**: The gripper does not approach the red coaster, or the bottle slips out during the moving process.
- **Level 1**: The drink bottle is placed outside the red coaster, the placement process disrupts the mug and causes it to topple, the robot gets stuck during rotation, the robot attempts placement but pours the water again, or the bottle is placed into the mug.
- **Level 2**: Only part of the drink bottle rests on the red coaster.
- **Level 3**: The drink bottle is fully and stably positioned on the red coaster.

Mouse Arrangement.

Step 1: Picking up the mouse.

- **Level 0**: The gripper does not move toward the mouse or moves around it without making contact.
- **Level 1**: The gripper approaches the correct grasping pose and touches the mouse but drops it after lifting it slightly.
- **Level 2**: The gripper pushes the mouse a significant distance before grasping it, the mouse is grasped but falls when lifted higher, or the gripper grasps the mouse in an unstable way.
- **Level 3**: The gripper successfully grasps the mouse without any slippage.

Step 2: Placing the mouse on the mouse pad.

- **Level 0**: The gripper remains stationary in the air and fails to move toward the mouse pad, releases the mouse from a high position and causes it to fall onto the table, or drags the mouse instead of lifting it.
- **Level 1**: The mouse is placed outside the mouse pad, the entire mouse lands on the pad but flips because it is released from a high height, or the gripper lifts the mouse again after placing it down.
- **Level 2**: Only part of the mouse is placed on the mouse pad, or the entire mouse is on the pad but bounces and shifts slightly because it is released from a relatively high height or because of the gripper retraction.
- **Level 3**: The gripper lowers to an appropriate height before releasing the mouse, ensuring that the entire mouse is securely placed on the pad.

Fold Towel.

Step 1: Grasping the left edge of the towel.

- **Level 0**: The gripper does not move toward the towel or moves around it without making contact.
- **Level 1**: The gripper moves toward the towel and attempts a grasping motion but fails to grasp any towel layer.
- **Level 2**: The gripper grasps only some of the towel layers, leaving others ungrasped, or grasps the towel substantially away from the edge.
- **Level 3**: The gripper successfully grasps all layers of the towel.

Step 2: Folding the towel to the right.

- **Level 0**: No folding motion toward the right is demonstrated.
- **Level 1**: After folding, the towel is left in a messy pile, such as being folded in thirds or bunched up, or the robot gets stuck during the folding process.
- **Level 2**: After folding, the overlap is off by more than one third, or the towel is folded in thirds with only a small top section.
- **Level 3**: After folding, the overlapping area exceeds two-thirds of the maximum possible overlap.

Unplug Charger.

Step 1: Grabbing the charger.

- **Level 0**: The gripper does not grab the charger.
- **Level 1**: The gripper grabs the charger but not tightly enough, resulting in failure to pull out the charger.
- **Level 2**: The gripper securely holds the charger, but there is a collision with the power strip during the process, though the charger is eventually pulled out.
- **Level 3**: The gripper securely holds the charger without colliding with other objects, and the charger is successfully pulled out afterward.

Variant family	Variant	Success rate (%)↑	Raw MSE (10^{-3})↓	CI-MSE (10^{-3})↓
Architecture	$\pi_{0.5}$	53.3	6.690	2.610
	X-VLA	27.5	5.873	2.770
	Gr00t N1.7	24.6	8.107	3.312
Data scale	20%	10.8	5.583	3.046
	40%	17.9	5.686	2.899
	60%	17.5	5.613	2.908
	80%	20.0	5.909	2.894
	100%	18.8	6.022	2.914
Training steps	20k	7.5	8.508	3.648
	40k	12.9	7.835	3.377
	60k	22.5	6.530	2.934
	80k	25.8	6.116	2.788
	100k	27.5	5.873	2.770
PEFT	All layers, rank 8	3.3	10.091	4.332
	All layers, rank 16	3.7	10.189	4.356
	QKV layers, rank 8	10.0	7.893	3.289
	QKV layers, rank 16	11.3	7.523	3.164
	QKV+FFN layers, rank 8	10.8	7.044	3.193
	QKV+FFN layers, rank 16	12.1	7.178	3.162
Action head	175M	17.5	7.938	3.232
	310M	25.8	6.449	2.943
	580M	18.8	6.954	2.905
	930M	28.7	6.595	2.891
	1220M	22.5	6.499	2.893
VLM backbone	florence2-base	4.6	8.621	3.772
	florence2-large	11.3	7.736	3.159
	paligemma2-3B-pt	7.1	7.729	3.441
	paligemma2-3B-mix	10.4	7.390	3.338

Table 6: Per-model rollout success rates and offline validation errors for the simulation experiments.

Step 2: Pulling out the charger.

- **Level 0**: The charger is not pulled out.
- **Level 1**: The gripper holds the charger in the air, or the charger slips from the gripper after being pulled out.
- **Level 2**: The gripper does not place the charger far enough to the right, or the charger is released from a relatively high position.
- **Level 3**: The charger is successfully pulled out, and the gripper places it to the right side of the power strip.

A.5 Per-Model Rollout Scores and Validation Errors

Table 6 shows the per-model rollout success rates and offline validation errors, including both raw MSE and CI-MSE, for the simulation experiments. Table 7 shows the per-model rollout scores, including Elo rankings and partial success scores, as well as offline validation errors for the real-world experiments.

A.6 Validation-Rollout Correlation by Inference-Time Methods

Our main experiments are carried out with temporal ensembling. Since RTC is also widely used in the community, we also study the correlation between validation error and rollout performance with RTC. We apply RTC to both real-world rollout and offline validation on the arrange-mouse task.

Comparing the rollout scores of the three inference-time methods, we find that temporal ensembling produces a different ranking from the other two methods. This supports the claim that inference-time methods can materially alter policy behavior. Comparing the validation rankings with the rollout rankings, we find that CI-MSE captures differences among inference-time methods and produces consistent rankings for the temporal ensembling and RTC groups. This suggests that applying the same inference-time method to offline validation can better match rollout-time behavior.

Task	Shift	Training pairs	Elo↑	Partial score↑	Raw MSE (10^{-3})↓	CI-MSE (10^{-3})↓
Pour water	Env.	4	1365.41	4.81	4.551	2.328
		8	1431.11	5.42	4.264	2.229
		16	1490.28	5.72	4.064	2.158
		32	1713.20	7.22	4.236	1.991
	Obj.	4	1301.72	3.83	3.556	1.997
		8	1430.44	5.28	3.063	1.969
		16	1521.38	5.83	2.815	1.946
		32	1746.46	7.38	2.772	1.911
Arrange mouse	Env.	4	1456.20	3.98	1.794	0.470
		8	1439.52	3.65	1.594	0.472
		16	1622.17	5.12	1.404	0.433
		32	1482.12	3.89	1.586	0.453
	Obj.	4	1479.75	3.11	1.931	0.613
		8	1524.31	3.65	2.520	0.676
		16	1659.96	4.55	2.628	0.647
		32	1335.99	2.00	2.755	0.749
Fold towel	Env.	4	1474.32	4.12	2.127	0.340
		8	1410.74	4.17	2.346	0.418
		16	1504.86	4.62	2.921	0.408
		32	1610.07	5.10	2.478	0.396
	Obj.	4	1477.20	3.24	2.793	0.418
		8	1524.19	3.66	2.694	0.368
		16	1503.32	3.38	2.942	0.371
		32	1495.29	3.42	2.569	0.412
Unplug	Env.	4	1250.28	1.09	3.097	2.988
		8	1534.27	3.66	3.353	3.330
		16	1607.27	4.84	3.423	3.026
		32	1608.18	4.59	3.228	3.198
	Obj.	4	1297.44	0.45	5.248	3.308
		8	1495.24	2.52	5.582	3.382
		16	1615.40	3.95	5.225	3.216
		32	1591.92	4.14	4.625	2.902

Table 7: Per-model rollout scores and offline validation errors for the real-world UMI experiments. Training pairs denotes the number of object-environment pairs used in the training data. Env. and Obj. denote cross-environment and cross-object evaluation settings.

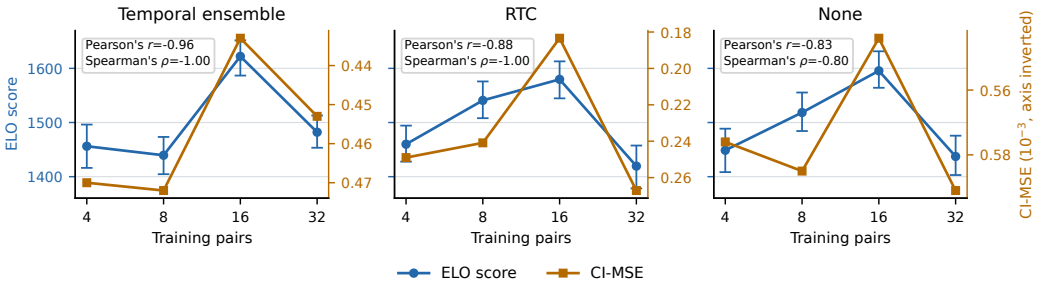


Figure 9: Validation error and rollout performance under different inference-time methods on the arrange-mouse task. Blue curves show Elo scores with 95% confidence intervals. Orange curves show CI-MSE on an inverted secondary axis to better show trend consistency with Elo scores.

Overall, CI-MSE achieves similar validation-rollout correlation under RTC and slightly lower correlation without any inference-time method. While this is only a single-task analysis, it suggests that the conclusions drawn under temporal ensembling are not purely an artifact of that specific inference-time method. Rather, the same validation principle appears to transfer to RTC when validation applies the corresponding rollout-time action processing. We therefore treat the temporal-ensembling results as the main experimental setting, while leaving a broader multi-task study of inference-time methods as future work.