REVISITING OPEN-LOOP EXECUTION IN ROBOTICS: TOWARD REACTIVE, HIGHER-PERFORMING POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Action chunking — the practice of predicting a sequence of actions and executing a prefix open-loop — has emerged as a key enabler of recent progress in imitation learning for robotic manipulation. However, executing long open-loop prefixes reduces reactivity, limiting policies' ability to correct for errors. Further, the mechanisms underlying these performance benefits remain poorly understood: prior works cite mitigating compounding errors, absorbing inference latency, or smoothing motions, but provide limited controlled evidence or guidance for preserving reactivity. In this work, we argue that long open-loop execution primarily helps short-context policies imitate "non-Markovian demonstrations". Across four simulation and two real-world tasks, we show that expert non-Markovianity shapes the relationship between task success and open-loop execution horizon. Further, we investigate the impact of compounding errors — the prevailing explanation for long open-loop execution in prior work — and find that while they matter, expert non-Markovianity has a much stronger impact in our experimental setting. Finally, we show that when policies are provided with a sufficiently long context in the high data regime, open-loop execution is no longer beneficial and the most reactive, closed-loop policies perform best. Overall, our findings motivate long-context, reactive policies as a more principled and performant paradigm. Website: https://open-loop-action-chunking.netlify.app/

1 INTRODUCTION

Imitation learning has recently seen rapid adoption in robotic manipulation. A key enabler of this progress has been *action chunking* — the practice of predicting a sequence, or "chunk," of actions and executing a fixed-length prefix open-loop before replanning from new observations. We term the length of this prefix the *execution horizon*. Popularized by ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2024), long execution horizons are now used broadly across imitation learning in both academia (Chi et al., 2024; Zhao et al., 2023) and industry (Zhao et al., 2024; Black et al., 2026; Physical Intelligence et al., 2025b; Figure AI, 2025; NVIDIA et al., 2025a). State-of-the-art works use open-loop execution horizons as long as 0.5 – 1 seconds (Chi et al., 2024).

However, executing longer open-loop segments and incorporating new observations less often comes at a cost: it reduces policy *reactivity* — the frequency at which the policy adjusts its behavior in response to execution errors or unexpected environmental changes. Reactivity has received comparatively little attention in literature even though uncertain dynamics, high-precision requirements, and dynamic environments can make frequent re-planning desirable. Further, as manipulation advances from slow, quasi-static tasks toward fast, dexterous, and dynamic contact-rich interaction, reactivity will only become more important. Human-level dexterity will require human-level reactivity.

This motivates the question: **why do long execution horizons appear so necessary?** In particular, even at a fixed prediction horizon, we ask why longer execution horizons still improve performance. Current justifications center around reducing the effective horizon of tasks to mitigate compounding errors (Zhao et al., 2023; Zhang et al., 2025), improving temporal consistency of actions to reduce unsmooth motions (Zhao et al., 2023; Chi et al., 2024), and absorbing delay from various inference latencies (Chi et al., 2024). However, these explanations have not been evaluated through extensive, controlled experimentation. Moreover, existing justifications explain why long execution horizons are beneficial but provide little guidance on how practitioners can preserve policy reactivity.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

An important fact about widely deployed imitation learning policies is that they use very short context lengths, often just the most recent 1 or 2 frames (Chi et al., 2024; TRI LBM Team et al., 2025; Kim et al., 2024; Zhao et al., 2023; Physical Intelligence et al., 2025b; Black et al., 2026). This can be insufficient for behavior cloning from human demonstrations: human experts may use implicit memory and additional information unavailable to the robot when selecting actions, making them *non-Markovian* with respect to the policy's observations. Our central claim is that **long execution horizons primarily compensate for policies' limited context lengths when trained to imitate non-Markovian experts**. We present a systematic empirical investigation of this claim across diverse tasks. We then show that increasing policy context length is an effective way to preserve reactivity while increasing task performance. Our findings do not argue against action-sequence *prediction*; rather, they suggest that open-loop long-sequence *execution* may not be needed.

Our key contributions are:

1. We show that non-Markovianity of the expert demonstrator is a key reason long execution horizons are necessary for short-context policies (Sec. 4.1).
2. We study the impact of compounding errors on execution horizon and find that expert non-Markovianity has a stronger influence on the optimal execution horizon in practice (Sec. 4.2).
3. We show that increasing context length reduces reliance on long execution horizons; sufficiently long context removes the benefit of long execution horizons; and increasing context length is a promising direction for training reactive policies with strong performance (Sec. 5).

Beyond these findings, we also introduce a simple double encoder architecture that improves long-context learning, and publicly release code (see reproducibility statement on page 10) for the automated Markovian and non-Markovian expert policies used in Sec. 4, supporting future studies on long-context imitation learning and the effects of expert properties.

2 Related Works

Action chunking in robotics: ACT (Zhao et al., 2023) and Diffusion Policy (Chi et al., 2024) popularized action chunking (in particular, long execution horizons) in robotic imitation learning. Long execution horizons are now common for both, specialist (Fu et al., 2024; Zhao et al., 2024; Yu et al., 2026) and generalist policies (TRI LBM Team et al., 2025; Black et al., 2026; Physical Intelligence et al., 2025b; Team et al., 2024; Kim et al., 2025; Liu et al., 2025a), and for models of all types, including flow/diffusion (Chi et al., 2024; Black et al., 2026; Physical Intelligence et al., 2025b), variational latent-variable (Zhao et al., 2023), and autoregressive (Pertsch et al., 2025). Across works, performance is often non-monotonic in the execution horizon: very short horizons underperform, intermediate horizons perform best, and very long horizons again underperform (Zhao et al., 2023; Chi et al., 2024). In this work, we explore this relationship between task success and execution horizon (which we call the ***success-horizon curve***) and study the factors that determine its shape. Intentionally using long execution horizons is largely unique to imitation learning (less prominent in reinforcement learning and model predictive control).

Current explanations for open-loop execution: Zhao et al. (2024) suggest that by reducing the frequency at which a policy re-plans, longer execution horizons "reduce the effective horizon" of tasks and thus help mitigate compounding errors. Zhang et al. (2025) provide theoretical and empirical support for this explanation. We test the practical importance of this in Sec. 4.2.

Practical justifications for longer execution horizons revolve around inference latency. They can allow policy roll-out to progress during latency (Chi et al., 2024; Black et al., 2025a) or smooth out motions when there are latency induced mis-alignments between new and previous chunks (Zhao et al., 2023; Chi et al., 2024; Black et al., 2025a; Malhotra et al., 2025; Liu et al., 2025b; Yu et al., 2026; Black et al., 2025b). While well-supported, neither explanation explains why long execution horizons are still useful in *simulated experiments*, where inference latency is abstracted away due to synchronous stepping of the simulator physics and policy. We focus on understanding why long execution horizons are required even under zero latency policies.

Many works have cited elements of history-dependence or partial observability such as "pauses in demonstrations" (Zhao et al., 2023), "temporally correlated confounders" (Zhao et al., 2023), or "mode commitment" (Chi et al., 2024), which (Block et al., 2023) unify under "non-Markovian expert demonstrations". Block et al. (2023) argue that chunked *prediction*, paired with generative

modeling, is necessary to capture non-Markovianity in expert demonstrations, but make no such argument about chunked *execution*. Zhang et al. (2025) acknowledge chunked execution may be beneficial when imitating non-Markovian experts but focus on studying compounding errors in their work. Concurrent work by Lazzati et al. (2026) agrees with our non-Markovianity hypothesis. While they investigate it through "delayed policies", we focus on varying the learner's context length and provide guidance on reactive policies. We expand on these topics in Sec. A.1.

Long-context imitation learning: We find that increasing context length is beneficial for policy reactivity. However, vast majority of imitation-learning policies condition on only a short history (of 1 or 2 frames), citing the *causal confusion* (de Haan et al., 2019). Recent work has pursued long-context imitation learning through two broad directions: directly scaling raw temporal context (Torne et al., 2025; Agarwal et al., 2026; Torne et al., 2026), and selectively compressing, retrieving, or maintaining task-relevant memory (Torne et al., 2026; Mark et al., 2026; Shah et al., 2026; Sridhar et al., 2025; Lei et al., 2026; Zheng et al., 2025; Shi et al., 2026; Li et al., 2025; Lin et al., 2026). We are interested in the first line of works because they retain maximum observability for inferring expert hidden state, which is important when imitating non-Markovian expert demonstrations.

3 PRELIMINARIES

3.1 IMITATION LEARNING AND DIFFUSION POLICY

We focus on behavior cloning, where a policy $\pi_\theta(A_t \mid O_t)$ approximates an expert conditional distribution $p_E(A_t \mid O_t)$ through supervised learning on expert observation-action pairs $\mathcal{D} = (O_t^{(i)}, A_t^{(i)})_{i=1}^n$. In particular, we use Diffusion Policies (Chi et al., 2024) (see Sec. C.1 for background).

Modern behavior cloning methods predict an action "chunk" $A_t = (a_t, a_{t+1}, \ldots, a_{t+T_p-1})$ from an observation history $O_t = (o_{t-T_o+1}, \ldots, o_t)$, where T_o is the **context length** and T_p the **prediction horizon**. As noted in Sec. 1, most works use short contexts ($T_o = 1$ or 2). Policies typically execute a prefix of this chunk open-loop before re-querying on new observations. We define the **execution horizon** T_{exec} as the number of actions executed before re-querying, and T_{exec}^* as the value of T_{exec} maximizing task performance. **We study how T_{exec} affects performance for fixed T_p.** In particular, while $T_{\text{exec}} = 1$ enables closed-loop reactive control, we ask why $T_{\text{exec}} > 1$ has generally been considered necessary for successful behavior cloning.

Throughout our experiments, we use the term ***success-horizon curve*** to denote the relationship between policy success rate and execution horizon (T_{exec}), plotted on a 2D curve, with task, dataset, policy class, and training procedure held constant. See Fig. 1 for an example.

3.2 NON-MARKOVIAN EXPERTS

We define a setup similar to Block et al. (2023). We treat the environment observations o_t as Markovian states for the environment[1]. Given o_t and a_t, the environment dynamics evolve as $p(o_{t+1} \mid o_{\leq t}, a_{\leq t}) = p(o_{t+1} \mid o_t, a_t)$ (Markovian). We define the following two forms for the expert E providing demonstrations in this environment:

- **Markovian Expert:** The expert's action distribution depends only on the current environment state: $\pi_E(a_t \mid o_{\leq t}) = \pi_E(a_t \mid o_t)$.
- **Non-Markovian Expert:** The expert's action distribution depends on both the current and past environment states: $\pi_E(a_t \mid o_{\leq t}) \neq \pi_E(a_t \mid o_t)$. We qualify experts which maintain a hidden state that carries distilled information from past observations in this category. For instance, experts providing demonstrations which require them to keep track of time.

Notably, human experts are inevitably non-Markovian: even when trying not to, they draw on earlier actions and observations, preferences, and unrecorded information when providing demonstrations.

[1]This approximation is commonly used for simplicity in algorithmic discussion in robotic behavior cloning (Physical Intelligence et al., 2025a). While not exactly true, it is often reasonable with sufficient camera views.

4 EXPERT MARKOVIANITY DOMINATES THE SUCCESS-HORIZON CURVE

We now demonstrate our first result: **the success-horizon curve is strongly influenced by Markovianity of the expert when training short-context policies.**

We compare the success-horizon curves of Diffusion Policies trained on Markovian and non-Markovian expert data across 4 diverse simulation benchmarks. *When the policy architecture, training procedure, and evaluation setting are fixed, we find that changing only the nature of the expert to Markovian shifts the curve toward near-monotonically decreasing with optimal execution horizon* $T^*_{exec} = 1$. We then address the compounding errors hypothesis — a common justification for long execution horizons from prior work — directly and show that, on the `FurnitureSimOneLeg` task (Heo et al., 2023), interventions designed to reduce compounding errors have a much smaller effect on the success-horizon curve than changing expert Markovianity.

In all experiments in this section, we train image-based Diffusion Policies with context length $T_o = 2$, prediction horizon $T_p = 15$, and execution horizons T_{exec} ranging from 1 to 15, holding all training and evaluation parameters constant besides the dataset.[2] We choose a variety of types of benchmarks, ranging from high-precision (`FurnitureSimOneLeg` (Heo et al., 2023), `GearInsertion`) to highly multimodal (`Kitchen` (Gupta et al., 2019)) and contact-rich (`Push-T` (Wei et al., 2025)). See Sec. B.1 for task descriptions and Fig. 9 for illustrations.

For `Push-T`, we use the ManiSkill simulator (Mu et al., 2021) for the expert comparisons in Sec. 4.1 (`PushT-M`) and Drake (Tedrake & the Drake Development Team, 2019) for the context-length experiments in Sec. 5 (`PushT-D`). ManiSkill's GPU parallelism makes training a Markovian RL expert more practical for Sec. 4.1. See Sec. B.1 for details.

4.1 POLICIES TRAINED ON MARKOVIAN EXPERT DEMONSTRATIONS EXHIBIT OPTIMAL EXECUTION HORIZON $T^*_{\text{EXEC}} = 1$

Firstly, we isolate the effect of expert Markovianity by testing whether replacing non-Markovian demonstrations with Markovian demonstrations shifts the optimal execution horizon to $T^*_{\text{exec}} = 1$.

4.1.1 EXPERIMENTAL SETUP

We develop two scripted experts for the `FurnitureSimOneLeg` task.

1. **Deterministic Markovian Expert:** A finite-state machine (FSM) that determines its state using the current and one previous environment states, and uses end-effector operational-space control and gripper PID control to move the robot toward FSM state-specific waypoints. Note that our policy conditions on 2 frames of history ($T_o = 2$), so a 2-Markovian expert is Markovian with respect to the policy's input.
2. **Non-Markovian Expert:** Constructed by injecting non-Markovian behaviors into the Markovian expert, including a hidden latent plan (consisting of episodically pre-determined waypoint offsets), sticky FSM state transitions (requiring the expert to stay in the current state for a latent number of steps (up to 16) before advancing), and latent-count alignment maneuvers (repeatedly noising the trajectory before picking or inserting the leg, for up to 10 timesteps per iteration and 3 iterations). These injections are designed to mimic human behaviors (e.g. pauses, delay in mentally registering subtask completion, sub-optimal or cyclic alignment motions).

In addition to these two scripted experts, we also compare a **human teleoperator** as an inherently non-Markovian expert.

We evaluate each expert at a variety of execution horizons and plot results in Fig. 1. We repeat the experiment for `PushT-M`, `GearInsertion`, and `Kitchen` in Fig. 2, plotting the success-horizon curve under **Markovian** and **human** experts. Further experimental details are in Sec. B.2.

4.1.2 RESULTS AND DISCUSSION

Expert Markovianity changes the success-horizon curve to monotonically decreasing: All policies trained on human expert data exhibit the inverted U-shape curve as commonly seen in the lit-

[2] We make an exception for `Kitchen`; See Sec. B.2.

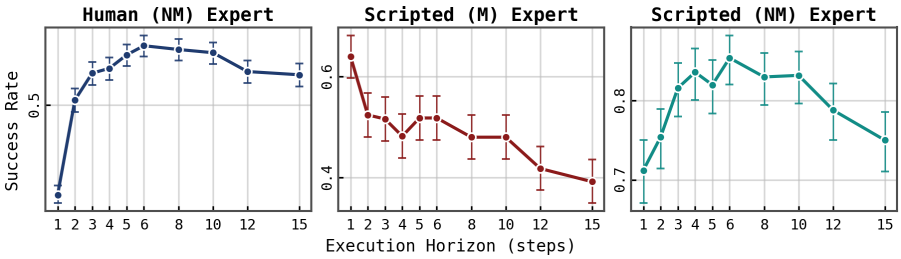


Figure 1: Success-horizon curves for Furniture-Sim `one-leg`. The policy trained on human non-Markovian (NM) data (**Left**) exhibits the typical inverted U-shape curve; the policy trained on Markovian (M) scripted data (**Center**) exhibits an approximately monotonically decreasing curve; adding non-Markovian behaviors to the scripted expert (**Right**) restores the inverted U-shape.

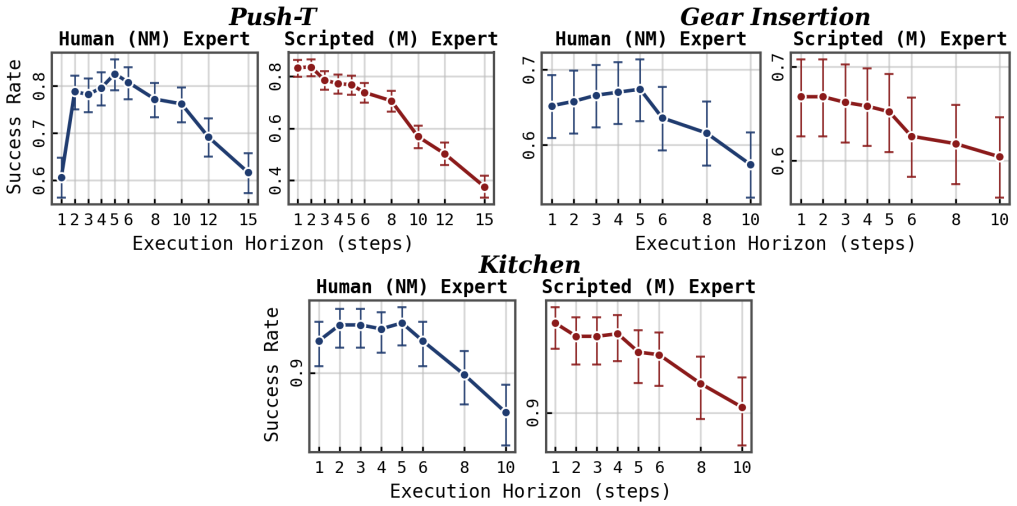


Figure 2: Success-horizon curve with non-Markovian (NM) human vs scripted Markovian (M) expert demonstrations for `PushT-M` (Top Left), `GearInsertion` (Top Right), and `Kitchen` (Bottom) tasks. All exhibit the typical inverted U-shaped curve when trained on human data, but all become monotonically decreasing when trained on Markovian expert data.

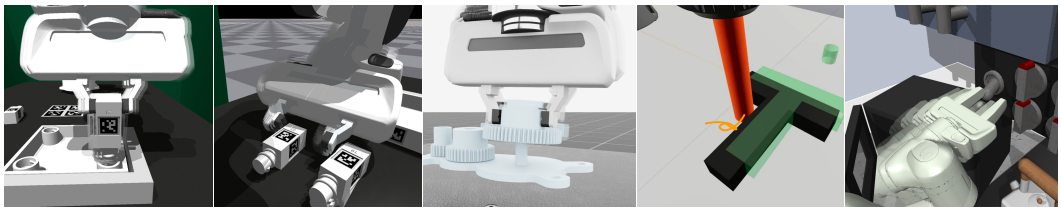
erature (Zhao et al., 2023; Chi et al., 2024). All policies trained on Markovian expert data exhibit optimal execution horizon of $T^*_{\text{exec}} = 1$ and near-monotonic decrease in success rate with T_{exec}. Given that the expert demonstrator was the only factor varied between the two sets of experiments, we determine that the nature of the Markovian expert's data is the driver of the shift in success-horizon curve in each benchmark task.

Notably, for `FurnitureSimOneLeg`, simply injecting non-Markovian behaviors into the Markovian scripted expert restores the inverted U-shape, showing that certain non-Markovian behaviors necessitate long execution horizons: non-Markovian features that make expert behavior depend on information especially difficult for the policy to infer from its limited context are most detrimental, such as sticky FSM transitions or latent-count alignment maneuvers, where the policy is effectively unable to observe the passage of time or the latent count (for example, while imitating a long pause, the policy is unable to determine how long it has been paused for based on 2 frames of context).

Short-horizon failures are consistent with our non-Markovianity hypothesis: Observing policy performance qualitatively, we find that dominant failure modes are caused by ambiguity in the desired action distribution when re-planning, caused by the expert being non-Markovian. With only a context length of 2, policies may select actions that appear valid based on their limited context, but are actually inconsistent with what the full-history conditioned expert would select. This can lead to mode-switching or unbounded repetition of locally valid motions.

For instance, in `FurnitureSimOneLeg`, we notice many failures around aligning to pick up the leg or aligning before insertion; the robot hovers and repeats valid approach maneuvers as demonstrated by the expert, but never commits to the pick or insertion. In `GearInsertion`, similar

failures occur when the robot aligns the gear to the shaft, rapidly switching between behaviors until failure. In `PushT`, because of multimodal contact possibilities, repeated mode-switching occurs almost anywhere around the T. In `Kitchen`, we observe the policy switch from one expert-consistent motion to another when very near two subtasks, ultimately completing neither. We visualize these failure modes in Fig. 3. Taken together, these behaviors point to expert non-Markovianity as the root cause: short-context policies cannot infer which of several locally valid expert behaviors is consistent with the entire trajectory history. Long execution horizons reduce this planning ambiguity by simply committing to a single mode for longer and re-planning less often. We discuss this mechanism further in Sec. D.1.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 3: Illustration of common failure modes of short-context policies using short execution horizons. **(a)**: `FurnitureSimOneLeg`: robot cycles between coherent alignment motions but fails to transition to insertion, causing timeout. **(b)**: `FurnitureSimOneLeg`: robot performs repeated alignment motions with the table leg, but fails to transition to grasp. **(c)**: `GearInsertion`: robot repeatedly aligns gear with shaft. **(d)**: `Push-T`: the robot jitters until timeout, oscillating between two modes (pushing the T rightward v/s moving around to push it upward). **(e)**: `Kitchen`: robot approaches light switch but abruptly changes course toward the top knob and collides.

So far we have only commented on how long execution horizons help address multimodality arising from the policy having limited context compared to the expert. However, the action distribution induced by a collection of expert demonstrations might itself be multimodal and thus could plausibly cause mode-switching in policy execution as well; while we don't explicitly study this case here, we discuss it in Sec. H.

Further, we discuss the benefits of policy reactivity in Sec. I.

4.2 Investigating the Role of Compounding Errors

Zhang et al. (2025) identify mitigating compounding errors (Sec. A.2) as a key function of long execution horizons. We test the practical importance of this on `FurnitureSimOneLeg` using two interventions designed to improve train–test state-distribution alignment. If compounding errors strongly determine the success-horizon curve, these interventions should disproportionately improve performance at short execution horizons and shift the optimum toward $T^*_{\text{exec}} = 1$. However, we observe only modest shifts compared with those caused by changing expert Markovianity.

Intervention 1: Data scale. We vary training dataset size. Although this does not directly align expert and policy distributions, broader state coverage is more likely to include states induced by the learned policy, mitigating compounding errors (Laskey et al., 2017; Chang et al., 2022).

In Fig. 4, we find that with only a 100 demonstrations, even the policy trained on deterministic Markovian expert data performs best at $T_{\text{exec}} = 4$. From 200 to 1,000 demonstrations, however, additional data produces only a slight tilt toward shorter execution horizons, indicating that compounding errors have comparatively modest influence once moderate state coverage is reached. Thus, our results support Zhang et al. (2025)'s hypothesis in the most data-limited setting, but suggest that, in the moderate-to-high data regime targeted by our main experiments, compounding errors are a secondary contributor with a substantially smaller effect than expert non-Markovianity.

Intervention 2: HG-DAgger. Next, we run 3 iterations of semi-automated human-gated DAgger (HG-DAgger) (Kelly et al., 2019) on our policies trained on 200 Markovian and non-Markovian expert demonstrations. For each DAgger iteration, for each T_{exec} independently, we roll out the base policy, record failed episodes, manually annotate an intervention timestep for each failure, reset the simulation to that timestep, and collect an expert correction from that timestep onward. We then fine-tune the policy on the additional correction data. See Sec. E for more details.

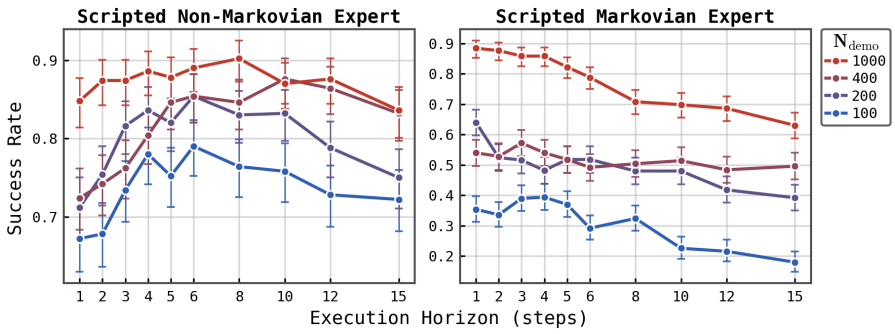


Figure 4: Success-horizon curves with $N_{\text{demo}} \in 100, 200, 400, 1000$ for non-Markovian (**Left**) and Markovian (**Right**) scripted experts. Only at $N_{\text{demo}} = 100$ does limited data substantially favor longer execution horizons; above 200 demonstrations, the curves shift only modestly.

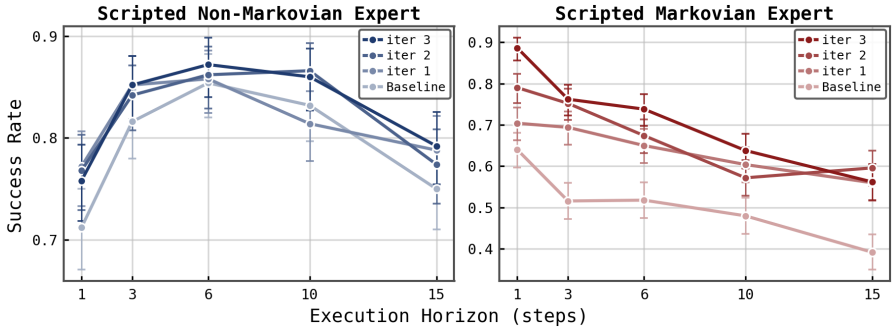


Figure 5: Success-horizon curves after applying HG-DAgger for Scripted Markovian (**Right**) and non-Markovian (**Left**) expert-trained policies. We find no noticeable shift for the non-Markovian policy, and a slight shift favoring lower execution horizons for the Markovian policy.

In Fig. 5, success rate increases over HG-DAgger iterations for both experts, yet the success-horizon curve shape largely remains the same, aside from a mildly stronger preference for shorter execution horizons in the Markovian case. Thus, we find compounding errors contribute minimally to the Markovian success-horizon curve, with their effects even less apparent under expert non-Markovianity. Qualitatively, successive HG-DAgger rounds primarily correct basic manipulation failures, such as missed insertions and screwing failures. For policies trained on non-Markovian expert data, however, HG-DAgger does not eliminate cyclic or idle behaviors, such as repeated alignment attempts during leg pick-up and insertion (see Figs. 3a and 3b). Because these corrections remain non-Markovian, HG-DAgger does not remove — and may reinforce — the ambiguity faced by short-context policies during re-planning that causes these cyclic failures.

Finally, we provide several supporting analyses in the appendix. We evaluate exploratory noise injection (Sec. F), a compounding errors mitigation technique proposed by Zhang et al. (2025). We also examine how proper checkpoint selection (Sec. J) and policy architecture (Sec. G.3) both demonstrate the impact of compounding errors while showing how it can be limited in practice.

Given that results in Sec. 4.1.2 are consistent across tasks, we expect the results in this section to readily extend to other tasks as well.

5 Long Context Reduces the Need for Long Execution Horizons

We now demonstrate that training policies with longer context lengths reduces reliance on long execution horizons and enables higher-performing *reactive* policies. In Sec. 4.1, we argue that long execution horizons mitigate the planning ambiguity that arises when short-context policies imitate non-Markovian experts. We thus hypothesize that increasing context length should shift T^*_{exec} toward 1. Experiments on 4 simulation tasks and 2 real-world tasks consistently support this hypothesis. At high data scales, long-context, short-execution-horizon policies even outperform short-context policies that rely on longer open-loop execution.

Agarwal et al. (2026) argue that naive context length scaling can be sufficient with appropriate architecture choice and large enough training dataset. We therefore adopt their cross-attention architecture and additionally apply a new method: adding separate image encoders for short- and long-range observations (see Sec. G.1 for details). Besides adopting a new model architecture, all other training and evaluation methodologies used in this section mimic those in Sec. 4.1.

For `FurnitureSimOneLeg`, we train Diffusion Policies with $T_0 \in \{2, 4, 8, 12, 16, 20\}$ on both human demonstrations ($N_{\text{demo}} = 200$) and scripted non-Markovian demonstrations ($N_{\text{demo}} \in \{200, 1000\}$). Fig. 6 (**Left**) shows the results for human demonstrations. At $T_o = 2$, an inverted U-shaped **success-horizon curve** is seen with $T^*_{\text{exec}} = 6$. At $T_o = 16$ and 20, however, the curve becomes nearly monotonically decreasing with optimal execution horizon $T^*_{\text{exec}} = 1$ or 2.

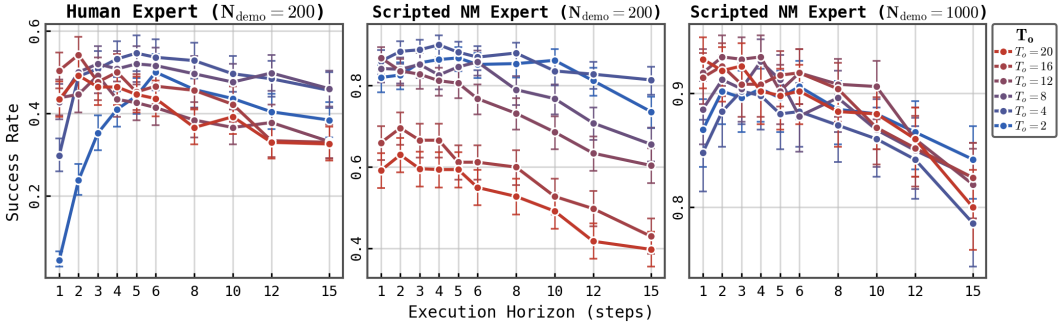


Figure 6: Success-horizon curves across context lengths for `FurnitureSimOneLeg`. Policies trained on 200 human demonstrations (**Left**) and 200/1000 non-Markovian scripted demonstrations (**Center/Right**). Increasing context shifts peak performance toward shorter execution horizons, while additional data disproportionately improves long-context policies at short horizons.

With non-Markovian demonstrations, at $N_{\text{demo}} = 200$ (Fig. 6 (**Left**)), the short context success-horizon curve is an inverted U-shape, while that at longer context lengths becomes roughly monotonically decreasing (consistent with human demos and our hypothesis). However, we see a drop in long-context policy performance, probably due to substantially expanded policy input space. However, increasing dataset size mitagates this; at $N_{\text{demo}} = 1000$ (Fig. 6 (**Right**)), long-context policies with short execution horizons out-perform short-context policies with longer execution horizons.

Fig. 7 shows that the trends above consistently replicate for policies trained on human-collected demonstrations across 3 other benchmark tasks (`PushT-D`, `GearInsertion`, and `Kitchen`).

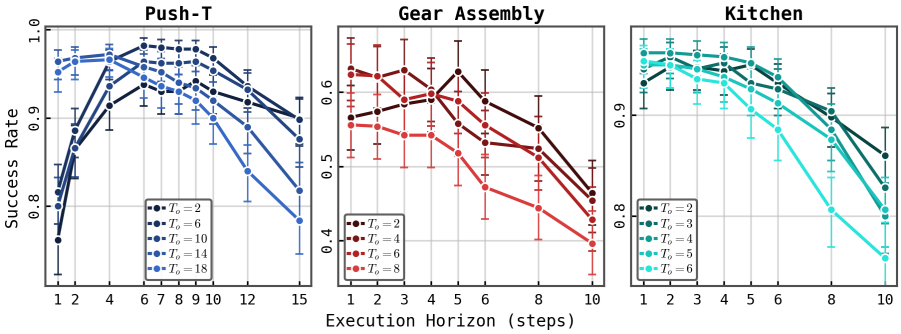


Figure 7: Success-horizon curves of policies trained across a range of context lengths for `PushT-D`, `GearInsertion`, and `Kitchen` on human demonstrations. With sufficiently long context length, $T_{\text{exec}} = 1$ becomes optimal or near-optimal in all tasks.

Lastly, we evaluate long-context policies on two real-world, highly-dexterous bimanual tasks: `SinglePillDispense` and `SlipIntoBaggie`, illustrated in Fig. 8. Both of these tasks require reactive dexterity — `SinglePillDispense` requires gentle tilting and intentional shaking to shift the pills toward the edge of the bottle without dumping out more than 1; `SlipIntoBaggie` requires careful shimmying to fit the paper slip into the baggie, which is barely larger than the paper. See Sec. B.3 for training and task details. We present results in Table 1 over a grid of T_o and T_{exec}.

As in the simulation experiments, increasing context from $T_o = 2$ to 8 shifts T^*_{exec} from 8 to 2 while improving performance. Reactivity is beneficial in both tasks: stopping dumping once a pill is dispensed in `SinglePillDispense`, and responding to unexpected paper or baggie deformation in `SlipIntoBaggie`. Meanwhile, longer context preserves the behavioral coherence that long execution horizons would otherwise provide. These results show that our findings generalize to real-world visual shifts, dynamics, and latencies: long context enables short execution horizons and outperforms short-context, long-horizon baselines.

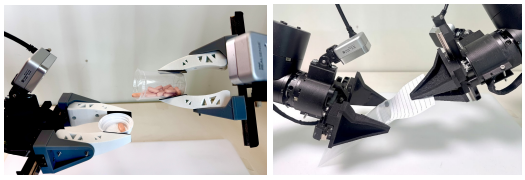


Figure 8: `SinglePillDispense` (Left) and `SlipIntoBaggie` (Right) tasks (real world). `SinglePillDispense`: Dump a single pill out of the bottle onto the cap and then right the bottle. `SlipIntoBaggie`: Fully insert the paper slip without crumpling into the baggie.

Table 1: `SinglePillDispense` and `SlipIntoBaggie` success rates (20 trials, 95% CI).

	`SinglePillDispense`			`SlipIntoBaggie`		
	$T_{\text{exec}} = 2$	$T_{\text{exec}} = 4$	$T_{\text{exec}} = 8$	$T_{\text{exec}} = 2$	$T_{\text{exec}} = 4$	$T_{\text{exec}} = 8$
$T_o = 2$	50% [29.9, 70.1]	60% [38.7, 78.1]	75% [53.1, 88.8]	60% [38.7, 78.1]	40% [21.9, 61.3]	80% [58.4, 91.9]
$T_o = 8$	**90% [69.9, 97.2]**	80% [58.4, 91.9]	60% [38.7, 78.1]	**85% [64.0, 94.8]**	70% [48.1, 85.5]	55% [34.2, 74.2]
$T_o = 12$	75% [53.1, 88.8]	70% [48.1, 85.5]	60% [38.7, 78.1]	75% [53.1, 88.8]	65% [43.3, 81.9]	65% [43.3, 81.9]

Finally, we note that performance declines at $T_o = 12$ in both `SinglePillDispense` and `SlipIntoBaggie`, reflecting a broader pattern across our experiments: once additional history saturates, the higher sample complexity of longer contexts degrades performance. At our data scales, this typically occurs only beyond the context length needed for reactive, low-execution-horizon policies to achieve peak performance, while higher data scales extend the range of beneficial context lengths. Determining the optimal tradeoff, and how it scales with data, remains future work.

Taken together, **these results establish long-context, reactive control as a viable and higher-performing alternative to long, open-loop execution** in the moderate-to-high-data regime.

6 Discussion and Conclusion

In this section, we conclude with additional discussion and directions for future research.

Firstly, **we emphasize that data scale is an important experimental variable**. We use 200+ demonstrations in a majority of our experiments. Our dataset size ablation illustrates why this matters: with only 100 demonstrations, even a policy imitating a fully observable, deterministic, Markovian expert requires non-trivial execution horizon, whereas this preference disappears as dataset size increases. Because datasets of 50 – 100 demonstrations remain common in robotics research, **we encourage evaluating results across dataset sizes to ensure results hold at scale.**

Secondly, we believe the prevailing approach to policy training — using short contexts with long open-loop execution — is poorly aligned with human behavior and responsive robotic operation. Long execution horizons were previously recommended to avoid causal confusion from history-conditioning (Zhao et al., 2023), but advances in data scale, compute, and policy architectures have since shifted this tradeoff. Recent work shows that even naively extending context is far less brittle than previously believed with appropriate architectures (Agarwal et al., 2026), alongside other developments in long-context learning (Torne et al., 2025; Mark et al., 2026; Sridhar et al., 2025; Torne et al., 2026). **We therefore encourage practitioners to train with longer contexts — even modestly longer at 4 – 8 frames — and evaluate correspondingly shorter execution horizons.**

Finally, we observe that current literature in long-context learning has focused on "global", observability and memory, like selecting the correct manipuland, counting repetitions, or recalling the location of hidden objects (i.e. in the "shell game" (Rhoda AI Team, 2026)). Accordingly, many methods represent observations via compacted summary tokens. Our results highlight a more subtle role of context that such summaries may not fully capture: by resolving hidden-state aliasing and enabling shorter execution horizons, **raw, visual context is critical for reactive, coherent, "local" control**. While semantic memory is important for long-horizon manipulation, we believe context-encoding methods should also be evaluated on dexterous tasks under short execution horizons.

AI USE STATEMENT

This work used generative AI for writing assistance, grammar, and formatting. While AI was used for assistance with writing code, including code that generated plots, all Figures are generated by human run code (AI was not used to directly generate figures). We have reviewed all code, training runs, and plots to reflect accuracy and take responsibility for all final content of this work.

REPRODUCIBILITY STATEMENT

We publicly release code at `https://anonymous.4open.science/r/Revisiting-Open-Loop-Execution-in-Robotics/README.md`. This can be used to reproduce our results as well as for further work on studying expert Markovianity's impact on imitation learning. Sec. B and Sec. C provide detailed information on tasks, training, and evaluation. Further, our project website `https://open-loop-action-chunking.netlify.app/` carries more information and visual evidence.

REFERENCES

Abhinav Agarwal, Adam Wei, Taylan Kargin, Michael Zeng, Cole Becker, Arif Kerem Dayi, Pablo Parrilo, Asuman Ozdaglar, and Russ Tedrake. Training and evaluating diffusion policies with long context lengths, 2026. URL https://arxiv.org/abs/2606.16447.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly, 2024. URL https://arxiv.org/abs/2404.03729.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies, 2025a. URL https://arxiv.org/abs/2506.07339.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking, 2025b. URL https://arxiv.org/abs/2512.05964.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior, 2023. URL https://arxiv.org/abs/2307.14619.

Jonathan D. Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data without great coverage, 2022. URL https://arxiv.org/abs/2106.03207.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning, 2019. URL https://arxiv.org/abs/1905.11979.

Figure AI. Helix accelerating real-world logistics. https://www.figure.ai/news/helix-logistics, February 2025. Published February 26, 2025; accessed July 13, 2026.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation, 2024. URL https://arxiv.org/abs/2401.02117.

Generalist Team. Gen-0: Embodied foundation models that scale with physical interaction. *Generalist AI Blog*, 2025. https://generalistai.com/blog/gen-0.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning, 2019. URL https://arxiv.org/abs/1910.11956.

Chengyang He, Xu Liu, Gadiel Sznaier Camps, Guillaume Sartoretti, and Mac Schwager. Demystifying diffusion policies: Action memorization and simple lookup table alternatives, 2025. URL https://arxiv.org/abs/2505.05787.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation, 2023. URL https://arxiv.org/abs/2305.12821.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts, 2019. URL https://arxiv.org/abs/1810.02890.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning, 2017. URL https://arxiv.org/abs/1703.09327.

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why does action chunking improve behavioral cloning performance in robotic control?, 2026. URL https://arxiv.org/abs/2608.02547.

Yuheng Lei, Zhixuan Liang, Hongyuan Zhang, and Ping Luo. Vpwem: Non-markovian visuomotor policy with working and episodic memory, 2026. URL https://arxiv.org/abs/2603.04910.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. Map-vla: Memory-augmented prompting for vision-language-action model in robotic manipulation, 2025. URL https://arxiv.org/abs/2511.09516.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models, 2026. URL https://arxiv.org/abs/2604.04161.

Min Lin, Xiwen Liang, Bingqian Lin, Jingzhi Liu, Zijian Jiao, Kehan Li, Ziang Yan, Yu Sun, Weijia Liufu, Yuhan Ma, Jiarui Hu, Yuecheng Liu, Shen Zhao, Yuzheng Zhuang, and Xiaodan Liang. Echovla: Robotic vision-language-action model with synergistic declarative memory for mobile manipulation, 2026. URL https://arxiv.org/abs/2511.18112.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation, 2025a. URL https://arxiv.org/abs/2410.07864.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling, 2025b. URL https://arxiv.org/abs/2408.17355.

Rhea Malhotra, Yuejiang Liu, and Chelsea Finn. Self-guided action diffusion, 2025. URL https://arxiv.org/abs/2508.12189.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021. URL https://arxiv.org/abs/2108.03298.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations, 2023. URL https://arxiv.org/abs/2310.17596.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. Bpp: Long-context robot imitation learning by focusing on key history frames, 2026. URL https://arxiv.org/abs/2602.15010.

Joseph Masterjohn, Damrong Guoy, John Shepherd, and Alejandro Castro. Velocity level approximation of pressure field contact patches, 2022. URL https://arxiv.org/abs/2110.04157.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations, 2021. URL https://arxiv.org/abs/2107.14483.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022. URL https://arxiv.org/abs/2203.12601.

Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, Ankur Handa, and Dieter Fox. Factory: Fast contact for robotic assembly, 2022. URL https://arxiv.org/abs/2205.03532.

Dantong Niu, Zhuoyang Liu, Zekai Wang, Boning Shao, Zhao-Heng Yin, Anirudh Pai, Yuvan Sharma, Stefano Saravalle, Ruijie Zheng, Jing Wang, Ryan Punamiya, Mengda Xu, Yuqi Xie, Yunfan Jiang, Letian Fu, Konstantinos Kallidromitis, Matteo Gioia, Junyi Zhang, Jiaxin Ge, Haiwen Feng, Fabio Galasso, Wei Zhan, David M. Chan, Yutong Bai, Roei Herzig, Jiahui Lei, Li Fei-Fei, Ken Goldberg, Jitendra Malik, Pieter Abbeel, Yuke Zhu, Danfei Xu, Linxi Fan, and Trevor Darrell. T-rex: Tactile-reactive dexterous manipulation, 2026. URL https://arxiv.org/abs/2606.17055.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025a. URL https://arxiv.org/abs/2503.14734.

NVIDIA, Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy

Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning, 2025b. URL https://arxiv.org/abs/2511.04831.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control, 2026. URL https://arxiv.org/abs/2512.01809.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025a. URL https://arxiv.org/abs/2511.14759.

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. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025b. URL https://arxiv.org/abs/2504.16054.

Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. Early stopping and non-parametric regression: An optimal data-dependent stopping rule, 2013. URL https://arxiv.org/abs/1306.3574.

Rhoda AI Team. Causal video models are data-efficient robot policy learners. *Rhoda AI Blog*, 2026.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks, 2025. URL https://arxiv.org/abs/2509.23224.

Rutav Shah, Yisu Li, Femi Bello, Yuke Zhu, and Roberto Martín-Martín. Memory retrieval in visuomotor policies for long-horizon robot control, 2026. URL https://arxiv.org/abs/2606.25136.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2026. URL https://arxiv.org/abs/2508.19236.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous, 2025. URL https://arxiv.org/abs/2503.09722.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025. URL https://arxiv.org/abs/2508.10104.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval, 2025. URL https://arxiv.org/abs/2510.20328.

Sunday Robotics. Act-2 preview: Generalizing reliability. *Sunday Robotics Blog*, jul 2026. URL https://www.sunday.ai/blog/act-2-preview.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. [Online]. Available: https://drake.mit.edu.

Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction, 2025. URL https://arxiv.org/abs/2505.09561.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models, 2026. URL https://arxiv.org/abs/2603.03596.

TRI LBM Team, Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, Naveen Kuppuswamy, Kuan-Hui Lee, Katherine Liu, Dale McConachie, Ian McMahon, Haruki Nishimura, Calder Phillips-Grafflin, Charles Richter, Paarth Shah, Krishnan Srinivasan, Blake Wulfe, Chen Xu, Mengchao Zhang, Alex Alspach, Maya Angeles, Kushal Arora, Vitor Campagnolo Guizilini, Alejandro Castro, Dian Chen, Ting-Sheng Chu, Sam Creasey, Sean Curtis, Richard Denitto, Emma Dixon, Eric Dusel, Matthew Ferreira, Aimee Goncalves, Grant Gould, Damrong Guoy, Swati Gupta, Xuchen Han, Kyle Hatch, Brendan Hathaway, Allison Henry, Hillel Hochsztein, Phoebe Horgan, Shun Iwase, Donovon Jackson, Siddharth Karamcheti, Sedrick Keh, Joseph Masterjohn, Jean Mercat, Patrick Miller, Paul Mitiguy, Tony Nguyen, Jeremy Nimmer, Yuki Noguchi, Reko Ong, Aykut Onol, Owen Pfannenstiehl, Richard Poyner, Leticia Priebe Mendes Rocha, Gordon Richardson, Christopher Rodriguez, Derick Seale, Michael Sherman, Mariah Smith-Jones, David Tago, Pavel Tokmakov, Matthew Tran, Basile Van Hoorick, Igor Vasiljevic, Sergey Zakharov, Mark Zolotas, Rares Ambrus, Kerri Fetzer-Borelli, Benjamin Burchfiel, Hadas Kress-Gazit, Siyuan Feng, Stacie Ford, and Russ Tedrake. A careful examination of large behavior models for multitask dexterous manipulation, 2025. URL https://arxiv.org/abs/2507.05331.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits: Adaptive execution horizons for robot policies, 2026a. URL https://arxiv.org/abs/2602.21445.

Mingxin Wang, Zhirun Yue, Renhao Lu, Yizhe Li, Zihan Wang, Guoping Pan, Kangkang Dong, Jun Cheng, Yi Cheng, and Houde Liu. Phaforce: Phase-scheduled visual-force policy learning with slow planning and fast correction for contact-rich manipulation, 2026b. URL https://arxiv.org/abs/2603.08342.

Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, and Russ Tedrake. Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels, 2025. URL https://arxiv.org/abs/2503.22634.

Yueyang Weng, Xiaopeng Zhang, Yongjin Mu, Yingcong Zhu, and Yanjie Li. Temporal action selection for action chunking, 2026. URL https://arxiv.org/abs/2511.04421.

Pengyuan Wu, Pingrui Zhang, Zhigang Wang, Dong Wang, Bin Zhao, and Xuelong Li. Closed-loop action chunks with dynamic corrections for training-free diffusion policy, 2026. URL https://arxiv.org/abs/2603.01953.

Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation, 2025. URL https://arxiv.org/abs/2503.02881.

Checheng Yu, Chonghao Sima, Gangcheng Jiang, Hai Zhang, Haoguang Mai, Hongyang Li, Huijie Wang, Jin Chen, Kaiyang Wu, Li Chen, Lirui Zhao, Modi Shi, Ping Luo, Qingwen Bu, Shijia Peng, Tianyu Li, and Yibo Yuan. χ_0: Resource-aware robust manipulation via taming distributional inconsistencies, 2026. URL https://arxiv.org/abs/2602.09021.

Thomas T. Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control, 2025. URL https://arxiv.org/abs/2507.09061.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Tony Z. Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity, 2024. URL https://arxiv.org/abs/2410.13126.

Yuchi Zhao, Miroslav Bogdanovic, Arjun Sohal, Liyu Tao, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Dynamic execution horizon prediction for chunk-based robot policies, 2026. URL https://arxiv.org/abs/2606.11408.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies, 2025. URL https://arxiv.org/abs/2412.10345.

A Additional Background

A.1 Additional Related Works

In Sec. 2 we comment in current justifications for longer execution horizons and action chunking. We give detailed explanation of works commenting on compounding errors and expert non-Markovianity in that section. We expand on some of these works and also comment on temporal consistency, smoothness, and variable execution horizons.

Other works commenting on expert non-Markovianity: Zhang et al. (2025) acknowledge that expert non-Markovianity can impact execution chunk length. However, they focus on studying compounding errors. They emphasize a result on the Robomimic Tool Hang task showing that long execution horizons remain beneficial even under a fully-observable, deterministic, Markovian expert[3]. We conduct experiments targeting both expert non-Markovianity and compounding errors, and conclude that expert non-Markovianity does play a strong role in the need for long execution horizons, and, in our experimental setting, a stronger role than compounding errors. Concurrent

[3]We find conflicting results in our experiments, but hypothesize that Zhang et al.'s result can be explained by the relatively small amount of training data used to train their learner (thus inflating compounding error), or by the low success rate ($\approx 70\%$) of the expert, causing the induced expert (which filters out the actual expert's failures) to be non-Markovian. See Sec. 4.2.

work by Lazzati et al. (2026) similarly explains the benefits of action chunking through improved capacity to imitate non-Markovian experts, alongside reduced compounding errors and "implicit ensembling." While they investigate non-Markovianity primarily through delayed policies, we study it by varying the learner's context length. These independent analyses strengthen the case that expert non-Markovianity is a key reason long execution horizons appear necessary.

Temporal consistency and smoothness: Many prior works have proposed increasing temporal consistency between consecutive action chunks (Zhao et al., 2023; Black et al., 2025a; Yu et al., 2026; Black et al., 2025b), with some works even claiming to reduce reliance on long execution horizons (Malhotra et al., 2025; Liu et al., 2025b). There are a variety of ways to do so, including averaging between chunks (Zhao et al., 2023; Yu et al., 2026), inference-time search (Liu et al., 2025b), inference-time guidance (Malhotra et al., 2025), or training or inference-time inpainting (Black et al., 2025a;b). In all cases, while increasing temporal consistency sometimes allows setting shorter chunk lengths, such methods inherently reduce reactivity by binding new actions to old actions, effectively trading execution horizon for a new "consistency" parameter. Moreover, averaging across chunks can be harmful under multimodal predictions, as it collapses distinct modes toward their mean.

Variable chunk-length policies: A recent line of work treats the execution horizon itself as adaptive, either by learning execution-length selectors (Weng et al., 2026; Zhao et al., 2026; Liang et al., 2026), or by using test-time confidence signals to truncate chunks and replan (Wang et al., 2026a). These methods can improve performance by trading off the shortcomings of long and short execution horizons at favorable times, but do not address why short-horizon policies fail in the first place. We target this root cause directly.

Another set of works trains low-frequency chunking base policies with high-frequency reactive heads (Niu et al., 2026; Xue et al., 2025; Sendai et al., 2025; Wu et al., 2026; Wang et al., 2026b). However, the reactive heads typically condition on a smaller set of observations and are bound to the outputs of the low-frequency base policies (i.e. by conditioning on latent action chunks (Xue et al., 2025) or using a partially denoised output from the base policy (Niu et al., 2026)), reducing them to only local refinement. We seek reactivity to even large environment changes that require full-policy re-planning.

A.2 Compounding Errors in Imitation Learning

Imitation learning policies are trained on states visited by the expert, but at test time, small policy errors can move the robot into states that are unlikely under the expert demonstrations. The policy is then forced to act on out-of-distribution observations, which can lead to further errors that compound over time (Ross et al., 2011). This phenomenon is known as *compounding errors* and is often used to motivate long execution horizons: executing more actions per policy query reduces the effective closed-loop decision horizon (Zhao et al., 2023; Simchowitz et al., 2025) and, consequently, the severity of compounding errors.

Data aggregation (DAgger (Ross et al., 2011)) is a standard method for reducing compounding errors: it iteratively rolls out the learned policy, queries the expert for the correct actions on states actually visited by that policy, aggregates these newly labeled states into the training dataset, and re-trains, thereby reducing mismatch between the training state distribution and policy-induced test-time state distribution.

We apply a variant of DAgger called HG-DAgger in Sec. 4.2 as an intervention to test whether long execution horizons primarily help by mitigating compounding errors.

B Experimental Setup

B.1 Benchmark Simulation Task Details

For `FurnitureSimOneLeg` (Heo et al., 2023), the goal of the task is to push the table top into the obstacle corner, pick up the table leg, then screw the table leg into the table top. The poses of the table top and table leg are randomized at the start of the episode using the "low" randomization setting provided by FurnitureSim (Heo et al., 2023). The scene has two cameras, a static front-facing camera

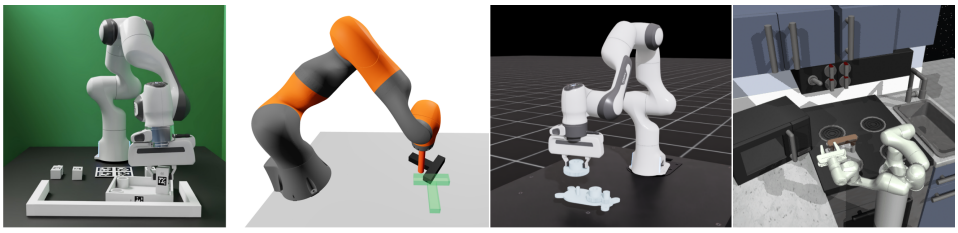


Figure 9: Benchmark Tasks. **`FurnitureSimOneLeg`** (Heo et al., 2023): pick up table leg and insert and screw into table top. **`Push-T-D`** (Wei et al., 2025): Push T-shaped slider to target pose, then return robot to reset pose. **`GearInsertion`**: Pick up large gear and insert onto gear carrier, meshed with other gears. **`Kitchen`** (Gupta et al., 2019): Complete any 4 subtasks (e.g., open microwave, move kettle, turn top/bottom knob, flip switch, open slide door, open hinge door), in any order; the dataset contains random subtask permutations. **All tasks use image-based observations.**

and a wrist-mounted camera. Proprioceptive state and action space and policy predicted-action frequencies for all tasks are summarized in Tables 2 and 3. Our expert comparison experiments (Fig. 1) use datasets of 200 demonstrations from each expert type. We use a SpaceMouse and adapt code from (Ankile et al., 2024) to collect human demonstrations.

In `Push-T`, the goal of the task is to push the T-shaped slider from a randomly initialized pose to a fixed goal pose. The robot is constrained to motion within the XY plane with a fixed end-effector Z coordinate and uses a cylindrical tool that allows only approximately point contacts with the T. We created a planar variant of the ManiSkill (Mu et al., 2021) `Push-T` environment, use the ManiSkill RL training pipeline to train our Markovian expert (see Sec. B.2 for more detail), and evaluate performance between Markovian and human experts (Fig. 2) in ManiSkill. However, for our `Push-T` long-context ablation (Fig. 7), we use the `Push-T` environment created in prior work by Wei et al. (2025) in Drake (Tedrake & the Drake Development Team, 2019) (as well as the open source dataset) provided by (Wei et al., 2025). The ManiSkill and Drake task variants are identical beyond simulation parameters except that the Drake implementation adds an additional requirement that the robot must return to its start position after pushing the T for success. The reason for the difference in simulators is that Drake provides a richer hydroelastic contact model (Masterjohn et al., 2022) that makes it well-suited for planar pushing, but training an RL expert is much more amenable in ManiSkill due to its GPU parallelism and simulation speed. Both scenes use two cameras, a static front-facing camera and a wrist-mounted camera. Our human vs Markovian expert comparison (Fig. 2) uses datasets of 300 demonstrations for each while our context length ablation (Fig. 7) uses 160 demonstrations.

For `GearInsertion`, the goal of the task is to pick up a gear initialized at a random pose and insert it onto a gear carrier, meshing it with the other gears. We use an adaptation of the "Factory Gear Mesh" task in IsaacLab (NVIDIA et al., 2025b; Narang et al., 2022). The scene has three cameras, a static front-facing camera, a static rear-left, upward-angled camera, and a wrist-mounted camera. The rear-left camera is added to improve visibility of the shaft and gear bore during insertion. All `GearInsertion` experiments (Figs. 2 and 7) use datasets of 200 demonstrations. The "human" expert dataset consists of 100 demonstrations collected using a SpaceMouse and 100 additional demonstrations synthetically generated using MimicGen (Mandlekar et al., 2023). The Markovian expert dataset consists of 200 successful rollouts from our Markovian scripted expert as described in Sec. B.2.

For `Kitchen`, the goal of the task is to complete any 4 subtasks, among 7 total: opening the microwave, moving the kettle to the far burner, turning the top or bottom knob, flipping the light switch, opening the left cupboard door, and sliding open the right cupboard door. All subtasks begin in their incomplete states. Success is determined when any 4 are completed. We use the environment from (Gupta et al., 2019) (with 2 added cameras — a scene camera and wrist camera) as well as the publicly released 581 human demonstrations (which we re-rendered with camera observations) combined with 581 Markovian scripted expert demonstrations that we generated for all experiments (Figs. 2 and 7).

Table 2: Benchmark Tasks – Proprioceptive Inputs and Action Spaces

Task	Proprioceptive State	Action Space
`FurnitureSimOneLeg`	16D: end-effector pose, velocity, and gripper width	10D: delta end-effector command (3D translation, 6D rotation, 1D gripper)
`Push-T`	3D: planar end-effector position and orientation	2D: delta end-effector position in the XY plane
`GearInsertion`	9D: 3D end-effector position, 4D orientation quaternion, 2D gripper state	7D: delta end-effector pose and gripper command
`Kitchen`	9D: seven arm-joint positions and two gripper-finger positions	9D: velocity commands for the seven arm joints and two gripper joints

Table 3: Benchmark Tasks – Policy Action Prediction Frequencies

Task	Predicted-Action Frequency
`FurnitureSimOneLeg`	10 Hz
`Push-T`	10 Hz
`GearInsertion`	30 Hz
`Kitchen`	12.5 Hz

B.2 Markovian Experts for Push-T, GearInsertion, and Kitchen

For `Push-T`, we train a Markovian Reinforcement Learning (RL) policy and evaluate it deterministically to generate Markovian expert data. The RL policy is parameterized as a Gaussian policy with an MLP and is trained from scratch using PPO (Schulman et al., 2017).

For `GearInsertion`, like for `FurnitureSimOneLeg`, we develop a Markovian scripted expert that uses a finite-state-machine (FSM) that determines its action using only the current environment state. For `GearInsertion`, in order to introduce some diversity during insertion (to prevent perfect insertions during all demonstrations, which yields a very thin training data manifold that the learner would easily step off of during rollout), we inject a small amount of Gaussian noise into both the executed and recorded actions proposed by the expert.

For `Kitchen`, we develop a deterministic Markovian scripted FSM-based expert for each of the 7 individual subtasks. During rollout, to complete sequences of 4 subtasks, we chain 4 of the subtask policies together with return-to-home primitives connecting them.

Because `Kitchen` permits completing any sequence of 4 of the 7 subtasks, the environment state alone does not uniquely specify which subtask the expert should execute next. To generate diverse Markovian expert demonstrations covering different subtask permutations, we sample a four-subtask sequence before each rollout and record it as an additional conditioning variable. Both the scripted expert and the Diffusion Policy have access to this sequence, making the expert's current and future subtask intent observable to the learner. Without this conditioning variable, randomly selecting a sequence would introduce latent expert state: identical environment states could produce different actions depending on a previously selected but unobserved subtask ordering. Note that this subtask sequence label is used only for the Markovian scripted-expert dataset and is not provided during human teleoperation.

To match `Kitchen`'s human teleop data collection environment (and the publicly released demonstration data), we also add random noise to the recorded observations during scripted expert rollout.

For all tasks, we roll out the Markovian expert and retain only successful episodes in the dataset. Conditioning on success may render the resulting induced expert non-Markovian; however, because our Markovian experts achieve high success rates (above 90%), we expect this effect to be limited.

For some tasks, we only show results for experts upto T_{exec} of length 10. This is done to focus the plots on the part of the curve with comparative differences.

B.3 Hardware Experiment Details

We train `SinglePillDispense` policies on just 1.34 hours of teleop data (516 demonstrations) and `SlipIntoBaggie` policies on 5.24 hours of data (540 demonstrations).

`SinglePillDispense`. The robot begins holding the pill bottle cap in its left gripper and pill bottle in its right gripper and must dispense exactly one pill into the cap within 30 s. Dispensing more or less than 1 pill constitutes failure. The time limit prevents incidental pill motion caused by prolonged, incoherent robot jitter from being counted as success. Each pill weighs approximately 0.6 g., and the bottle contains up to 24 pills and measures 1.7 in. diameter by 2.8 in. length.

`SlipIntoBaggie`. The robot must pick up a plastic baggie from the table and a paper slip from an elevated surface, then insert the slip into the baggie within 2 min. Success requires the slip to remain uncreased; an episode is terminated and marked as failure if the slip is creased or actively being creased. The slip measures 83.7×73.8 mm (± 2 mm), and the baggie measures 127×76.2 mm.

Hardware and control implementation. Both tasks use two ARX-5 follower arms with custom grippers, teleoperated using two ARX-5 leader arms. Observations comprise RGB images from two wrist-mounted Intel RealSense D405 cameras and one overhead Intel RealSense D435 camera. We use the same training procedure as for the simulation tasks and train Diffusion Policies with delta end-effector-pose actions and 10 Hz actions and observations. A 125 Hz linear interpolator generates intermediate target poses between the policy-predicted 10 Hz actions, which are converted to target joint positions by an inverse-kinematics solver and executed by a low-level joint-position PD controller. During evaluation, we apply two very simple but effective latency-control mechanisms. Firstly, we cache observation embeddings for future inference calls to re-use, preventing redundant forward passes of the observation encoder. Secondly, to avoid discontinuities between action chunks caused by inference latency, as discussed in Sec. 2, we linearly interpolate from the final action of the previous chunk to the action 180 ms into the newly predicted chunk before executing the remainder of the new chunk (the value of 180 ms was tuned empirically).

Evaluation procedure. During training, we save the 5 checkpoints with lowest DDIM MSE loss on the validation set. We screen each checkpoint using a small set of preliminary hardware rollouts, considering both task performance and qualitative stability. We then evaluate the chosen checkpoint on an independent set of 20 additional trials. We note that checkpoint selection was particularly important at long context lengths – we discuss this further in Sec. J.

Note on lighting variation. We note that color-jitter augmentations were *unintentionally* omitted during training. To limit lighting variation, all evaluations were conducted within the same five-hour window each day. Nevertheless, demonstrations were collected under varying lighting conditions, potentially adding extra visual variation into the dataset and reducing the effectiveness of some data samples. We expect that re-training with stronger lighting invariance would improve performance, particularly for longer-context policies that are already more subject to out-of-distribution inputs.

C Training and Evaluation Methods

C.1 Diffusion Policy

Diffusion Policy (Chi et al., 2024) represents $\pi_\theta(A_t \mid O_t)$ as a conditional denoising diffusion model over action chunks. We parameterize this distribution with a neural network ϵ_θ comprising image encoders that map the observation history O_t to observation features and a conditional U-Net that conditions on these observation features and iteratively denoises the action chunk.

During training, a clean expert chunk A_t is corrupted as

$$A_t^k = \alpha_k A_t + \sigma_k \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

where k indexes the diffusion noise level. The policy learns a family of denoisers, one for each noise level, typically by predicting the added noise:

$$\min_{\theta} \mathbb{E}_{(O_t, A_t)\sim\mathcal{D},;k,;\epsilon} \left[\left| \epsilon - \epsilon_\theta(A_t^k, k, O_t) \right|^2 \right].$$

At inference, the policy initializes $A_t^K \sim \mathcal{N}(0, I)$ and iteratively denoises it to produce an action chunk $\hat{A}_t$. The robot executes the first T_{exec} actions open-loop before re-querying the policy. Since actions are discretized at fixed interval dt, the corresponding open-loop execution time is $T_{\text{exec}}dt$. Note that, throughout this work, we fix dt and use T_{exec} synonymously with open-loop execution time.

C.2 Training and Evaluation Parameters

In all simulation experiments, we use ResNet-18 (He et al., 2015) as the image encoder and fine-tune its weights during training. For `FurnitureSimOneLeg`, `GearInsertion`, and `Kitchen`, we initialize the encoder with R3M-pretrained weights (Nair et al., 2022). We use R3M because the ablations in (Ankile et al., 2024) show that it outperforms ImageNet-1K, CLIP, and other initialization methods on FurnitureSim tasks. For `Push-T`, we instead use the ImageNet-1K-pretrained weights provided by robomimic (Mandlekar et al., 2021) and replace the encoder's BatchNorm layers with GroupNorm, following Diffusion Policy (Chi et al., 2024). For the hardware experiments, `SinglePillDispense` and `SlipIntoBaggie`, we use a larger, pre-trained DINOv3 (Siméoni et al., 2025) vision transformer and fine-tune during training with $\frac{1}{10}$ of the policy learning rate. We find superior performance with DINOv3 (compared to CLIP or ResNet-based image encoders) in high precision real-world tasks.

For inputs containing C cameras and T_o observation timesteps, we use C separate image encoders, one for each camera, and share each encoder's weights across the corresponding T_o timesteps for training efficiency. All remaining shared training and model parameters are reported in Tables 4 and 5.

Table 4: Training Hyperparameters Shared for All Models

Parameter	Value
Max LR	10^{-4}
LR Scheduler	Cosine
Weight Decay	10^{-6}
Warmup Steps	500
Batch Size	64
Image Size Input	$N_{\text{cameras}} \times 320 \times 240 \times 3$
Image Size Encoder	$N_{\text{cameras}} \times 224 \times 224 \times 3$
Image Augmentations	95% random crop
Encoder Feature Projection Dim	128

Table 5: Diffusion Model Hyperparameters

Parameter	Value
U-Net Down dims	$[256, 512, 1024]$
U-Net Parameters	69 million
DDIM Inference Steps	10

For each dataset, we reserve 5% of demonstrations for validation (except in `SinglePillDispense` and `SlipIntoBaggie`, where we reserve 2%); thus, the effective training-set size is $0.95\times$ ($0.98\times$ for `SinglePillDispense` and `SlipIntoBaggie`) the stated number of demonstrations for each task. We intentionally train each policy well into the

overfitting regime, where validation loss has increased substantially beyond its minimum, because Diffusion Policies have often been found to achieve their best rollout performance in this regime (He et al., 2025).

During training, we save checkpoints every 5 epochs. We then select the 5 checkpoints with the lowest DDIM MSE loss on the validation set for closed-loop evaluation; we do not perform closed-loop evaluations during training. For each of the 5 chosen checkpoints and each execution horizon, we run 500 evaluation episodes, terminating each episode upon success or timeout. Identical environment random seeds were used across all simulation evaluations except in `GearInsertion`. Given the large number of evaluation trials (500 per setting) and the consistency of our findings across benchmarks, we do not expect this difference to affect our conclusions. We set each task's timeout to approximately 120% of the longest demonstration in its dataset. Error bars throughout the paper denote 95% Wilson confidence intervals for the success probability of the reported (highest-performing) checkpoint for each task, architecture, and execution-horizon configuration; they characterize uncertainty from finite evaluation rollouts, but do not capture randomness from the training process. A more rigorous analysis with explicit hypothesis testing and correction for multiple policy comparisons as in (TRI LBM Team et al., 2025) is desirable but left to future work.

D Understanding the Benefits of Long Execution Horizons and Long Context

D.1 Why Long Execution Horizons Prevent Failure

In this section, we provide informal conceptual explanations for understanding why long execution horizons prevent failure in the presence of non-Markovian demonstrations. For this section specifically, subscripts indicate time index, and superscripts are used to distinguish states distinct in task space.

Consider the phenomenon of *hidden-state aliasing*: at a given state, conditioned on full history, the expert has a specific action distribution. However, the policy's short history may be consistent with multiple full histories in the training dataset that induce different expert action distributions. Even a well-trained short-context imitator may therefore sample an action from any one of these valid-appearing distributions and select an action inconsistent with the expert's full-history-conditioned behavior. This can send the policy down an incorrect path or into a cycle. We illustrate this phenomenon with an example in Fig. 10.

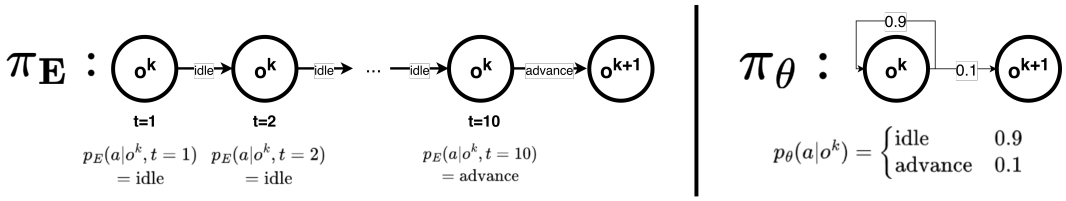


Figure 10: Illustration of hidden-state aliasing in an idle demonstration. While making high precision maneuvers in tasks like `Push-T`, humans often pause to align, observe, and think. Such pauses show up as a sequence of idle actions $o_1^k \to o_2^k \to \cdots \to o_{10}^k$ in the training data. In this example, the full-history conditioned expert π_E advances after it observes the 10th idle state. The imitator π_θ only observes the current state o_0 and cannot infer idle count, so both idling and advancing appear valid. The imitator is therefore unlikely to reproduce the expert's exact sequence of 10 idle actions before advancing, and may idle for much longer than desired, resulting in task timeout.

Long open-loop execution reduces the likelihood of incorrect action selection from hidden-state aliasing by reducing the probability of the policy re-planning at an aliased state. In particular, a policy with execution horizon T_{exec} re-plans every T_{exec} timesteps; assuming no correlation between phase with state, any aliased state has probability $\frac{1}{T_{\text{exec}}}$ of coinciding with a re-planning step. In the example in Fig. 10, the policy can assuredly imitate the 10 idle actions if it executes the entire 10-step aliased sequence open-loop, without re-planning (for instance, if policy inference happens

in the state right before o^k and execution horizon is at least the length of the idle interval). If the policy re-plans at any intermediate step, it may wrongly predict to idle for too long or too short.

For mostly unimodal tasks like `FurnitureSimOneLeg` or `GearInsertion`, there may not necessarily be aliased states where some apparently valid action distributions lead to unrecoverable outcomes. In most cases, hidden-state aliasing is detrimental by inducing unintended *cycles* — any sequence of states $o_0^k \to o_1^{k+1} \to o_2^{k+2} \to \cdots \to o_T^k$ that revisits a state — that can lead to task timeout. In particular, by revisiting the aliased state o^k multiple times, the policy may repeatedly select the locally-valid cycling action, keeping it in the cycle indefinitely. The simplest example of such a cycle would be an idle demonstration (i.e. Fig. 10), where a policy that over-predicts idling may idle forever. We illustrate another example of cycling in Fig. 3. Note that even when cycles do not exist in any single demonstration, they may appear in the cumulative dataset if the expert is non-Markovian, i.e. if one demonstration contains $o_t^k \to \cdots \to o_{t+i_1}^{k+1}$, and another contains $o_t^{k+1} \to \cdots \to o_{t+i_2}^k$.

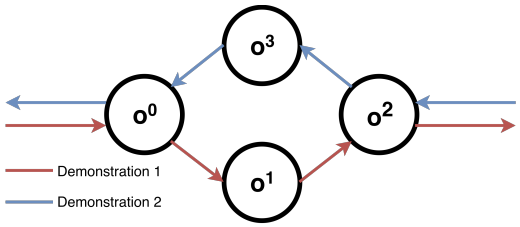


Figure 11: Illustration of cycling. Two demonstrations jointly induce a cycle in the aggregated dataset. When observing only o^0 or o^2, a short-context imitator cannot determine whether to continue or exit the cycle, since both actions appear valid in the training data. It may therefore repeatedly switch between the demonstrated trajectories and enter a cycle: $o^0 \to o^1 \to o^2 \to o^3 \to o^0$. Longer execution horizons reduce the chance of entering such a cycle: a chunk of actions originating at o^0 toward o^1 and executed open-loop with $T_{exec} \geq 3$ will cross beyond o^2, whereas re-planning at o^2 can increase the chance of returning to o^3 and entering a cycle.

Crucially, using an execution horizon of T_{exec} grants immunity from getting trapped in cycles of length $< T_{\text{exec}}$. Because the policy executes an entire T_{exec}-step action chunk open-loop, it traverses past the aliased state o^k in one open-loop chunk and therefore cannot re-select the cyclic behavior. For cycles longer than T_{exec}, increasing T_{exec} still reduces probability of cycling by simply decreasing the probability of re-planning on the aliased state to $\frac{1}{T_{\text{exec}}}$.

While the probability of repeating a cycle many times is theoretically very small, Diffusion Policies are known to overfit and learn high-Lipschitz functions rather than learning the true multimodal expert distribution (Pan et al., 2026). In practice, when a policy repeats a cycle once, we find that it is likely to repeat it many times.

In summary: long open-loop execution can be thought of as a mechanism to reduce the probability of incorrect action selection from hidden-state aliasing and to prevent unbounded cycling.

D.2 Why Long Context Lengths are a Principled Substitute for Long Execution Horizons

In this section, we argue that long context lengths can serve the same role as long execution horizons when imitating non-Markovian demonstrations and, when they contain sufficient history, do so more directly. As established in Sec. D.1, long execution horizons reduce the probability of hidden-state aliasing by reducing the frequency the policy re-plans at aliased states. Longer contexts instead preserve frequent re-planning while using preceding observations to disambiguate the expert's intended behavior at those aliased states. We illustrate this in Fig. 12.

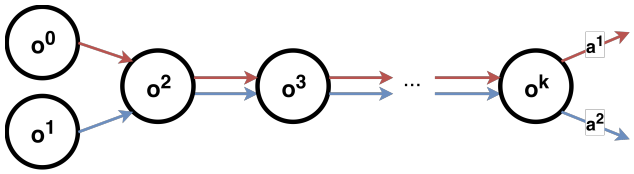


Figure 12: Illustration of how context length disambiguates the expert action distribution. The observations $o^2, \ldots, o^k$ form an aliased sequence: conditioned only on observations within this sequence, a short-context policy cannot identify the expert's intended behavior (whether to imitate demonstration 1 or 2). A long execution horizon reduces the probability of re-planning within the sequence, but ambiguity remains whenever re-planning occurs there. By contrast, a sufficiently long context includes the preceding observation o^0, which disambiguates the correct expert action (a^1) distribution throughout the sequence, regardless of re-planning phase.

The same distinction applies to cycling. Long execution horizons reduce the opportunity to re-select a cyclic action, while a context spanning the relevant history can reveal that the aliased state has already been visited and disambiguate the expert's exit behavior.

In summary, long contexts address the same failure modes as long execution horizons, but do so in an arguably more principled way (that also maintains policy reactivity), by eliminating the underlying planning ambiguity rather than merely reducing how often the policy encounters it.

E HG-DAgger Implementation Details

We provide additional implementation details for the HG-DAgger experiment in Sec. 4.2. For each expert type, all execution horizons were initially evaluated with the same base policy, but HG-DAgger was thereafter run independently for each execution horizon. Each execution horizon-branch maintained its own aggregated dataset, checkpoints, failure rollouts, and correction trajectories; correction data were never shared across execution horizons.

In each round, we rolled out the best-performing checkpoint from the preceding round until obtaining 50 failed episodes. A human annotator labeled the *gating timestep* of each failure, defined as the first timestep at which the policy deviated from a nominal trajectory in a manner that ultimately caused failure. For policies trained on non-Markovian expert data, the annotator also labeled the finite-state-machine state from which to initialize the scripted expert. The expert was then rolled out from the labeled state until success; episodes were discarded after 10 consecutive failed correction attempts. Provided at least 40 corrections succeeded, we added 40 to the corresponding branch's dataset, retrained its policy for 50,000 steps, and evaluated the resulting checkpoints before the next round. We chose to conduct 3 rounds as this was sufficient for convergence for the experiments in (Kelly et al., 2019) and allows us to assess early trends in the success-horizon curve, when the effects of DAgger are greatest.

Because HG-DAgger must be run separately for each value of T_{exec}, we only evaluated 5 values of T_{exec}, instead of the $T_{\text{exec}} \in \{1, 2, 3, 4, 5, 6, 8, 10, 12, 15\}$ sweep used in other experiments. We note that, with 3 rounds of HG-DAgger, 50 failed rollouts collected at each round, for 5 values of T_{exec}, for both the Markovian and non-Markovian expert: a total of 1500 episodes had to be manually annotated, by a single annotator for consistency.

F Exploratory Noise-Injection Experiment

Expanding on Sec. 4.2, we evaluate one more intervention designed to mitigate compounding errors proposed by Zhang et al. (2025): exploratory noise injection; and we show it's impact on the success-horizon curve.

Exploratory noise injection mitigates compounding errors by injecting noise during expert data collection to expand supervision around the expert distribution. We mimic the formulation in (Zhang et al., 2025), where a fraction α of trajectories are collected cleanly, while the remaining trajectories

execute perturbed expert actions $u_t = \pi^\star(x_t) + \sigma_u z_t$ but record the clean expert label $\pi^\star(x_t)$.[4] In our experiment, we treat σ_u as a scalar, multiplied to state-dependent noise standard deviations, because different states require different levels of precision. We apply this technique during data collection using our non-Markovian scripted expert. While Zhang et al.'s theoretical analysis for noise injection require deterministic Markovian experts, we discuss caveats with using a non-Markovian expert below.

We sweep $\sigma_u \in \{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$ and $\alpha \in \{0.6, 0.7, 0.8, 0.9\}$ and show results in Fig. 13.

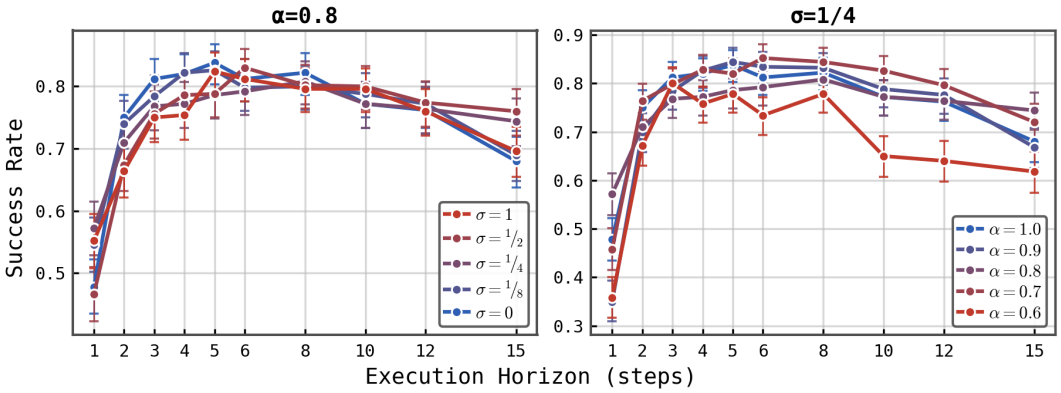


Figure 13: Success-horizon curves of policies trained on scripted non-Markovian expert data under exploratory noise injection. Left: ablation over the action-noise scale σ_u, with fraction of clean trajectories fixed at $\alpha = 0.8$. **Right**: ablation over clean-trajectory fraction α, with $\sigma_u = \frac{1}{4}$. We use $T_o = 1$ in this experiment for reasons described below.

Surprisingly, moderate α and σ_u values improve success rates at both very low $T_{\text{exec}} = 1$ and at very high $T_{\text{exec}} = 8, 10, 12, 15$. Further, poor selection of α and σ_u degrades performance: small α leaves too few clean trajectories, while insufficient or excessive action noise σ_u provides less useful corrective supervision. The experiment doesn't show any clear shift of the success-horizon curve preferring lower execution horizons, but we also discuss caveats with this result in the coming paragraphs.

First, because we use a non-Markovian expert in this experiment, we must use a single-frame context $T_o = 1$ and remove proprioceptive velocity from the observation space. Velocity and multi-frame observations typically provide strong signals of the expert's intended direction of motion. Under noise injection, however, these signals also encode the applied perturbations, which can obscure the expert's latent intent and conflate reduced observability with compounding-error mitigation. This caveat does not bias the result, but explains why we use $T_o = 1$ in this experiment.

Second, even with a single-frame context, noise injection may increase multimodality. Our non-Markovian scripted expert uses an FSM with latched state transitions: if a perturbation moves the system back across a transition boundary immediately after the FSM advances, the FSM remains in its new state. Similar physical states may therefore receive actions associated with different FSM states, introducing additional target ambiguity. Because the applied perturbations are small relative to the expert's commanded actions (i.e. 0.005 m standard deviation noise at the highest $\sigma_u = 1$ versus 0.025 m delta actions), we expect this effect to be limited, but it remains relevant when interpreting Fig. 13.

Third, the state-action-sequence mismatch described in Footnote 4 may disproportionately affect policies with long execution horizons. Although all policies predict multi-step action chunks, long-execution-horizon policies rely more heavily on later actions in each chunk, where this mismatch is greatest. This artifact therefore biases the comparison toward preferring shorter execution horizons.

[4] As noted by Zhang et al., this intervention has an important caveat in chunked prediction. For one-step behavior cloning, every perturbed state is paired with a corrective expert action. With multi-step prediction, however, later expert action labels are evaluated at states reached under noisy actions rather than under the preceding clean action labels in the target chunk. This state–action-sequence mismatch can make useful exploratory coverage appear as target uncertainty, particularly for long prediction horizons.

These caveats introduce opposing biases: additional multimodality may favor long execution horizons, whereas state-action-sequence mismatch may favor short execution horizons. We therefore do not treat this experiment as a clean test of whether exploratory noise injection reduces the need for long open-loop execution. Nevertheless, it provides a direct comparison with a complementary compounding-error mitigation technique proposed by Zhang et al. (2025). Within this setting, exploratory noise injection improves absolute performance for some hyperparameter choices, but provides no clear evidence of systematically reshaping the success-horizon curve toward shorter execution horizons.

G Additional Details on Long-Context Training

G.1 Long-Context Architectures for Tasks and Double Encoder Method Details

Contemporary techniques for training policies that condition on a large number of raw image observations are still evolving. In this section, we propose new heuristics related to visual encoding to improve long-context performance.

For `FurnitureSimOneLeg`, `GearInsertion`, and the two real-world tasks, following (Agarwal et al., 2026), we use a U-Net Diffusion Policy architecture with cross-attention conditioning. We also apply a new method we call the "double encoder", where the policy maintains separate image encoders (with separate parameters) for short-range and long-range observations. The most recent `short_range` images are encoded by both encoders, and all earlier frames are encoded only by the long-range encoder; the U-Net conditions on all features via cross-attention. Intuitively, the double encoder allows the policy to learn separate representations for immediate control versus long-range planning and inferring expert hidden state. To prevent the policy from relying exclusively on the short-range pathway and ignoring long-range features, we also apply a dropout on the short-range encoder output, replacing it with a learned "null" feature vector with some probability during training. In practice, we use `short_range = 2` and a dropout rate of 0.3 for all policies. Although the double encoder is a heuristic architectural choice, we find that it stabilizes the performance of the long-context policies, enables effective recall of semantic information even with very long context lengths, and gives more predictable performance results (see Sec. G.2). We now describe the architecture in detail.

The U-Net and cross-attention conditioning architecture implementations are taken from (Agarwal et al., 2026).

The double encoder is original work. It instantiates two *architecturally identical but independently parameterized* observation encoders: a long-range encoder E_{L} and a short-range encoder E_{S}. The long-range encoder is applied to all T_o frames, producing long-range tokens $\mathbf{z}_t^{\mathrm{L}} = E_{\mathrm{L}}(\mathbf{o}_t)$ for $t = 0, \ldots, T_o - 1$. The short-range encoder is applied only to the most recent T_s frames ($T_s \leq T_o$), producing short-range tokens $\mathbf{z}_t^{\mathrm{S}} = E_{\mathrm{S}}(\mathbf{o}_t)$ for $t = T_o - T_s, \ldots, T_o - 1$. The two streams are concatenated into a single conditioning sequence of $T_o + T_s$ tokens that the denoising network conditions on via cross-attention. Consequently, the most recent T_s frames are represented *twice* — once through the shared long-horizon pathway and once through the short-horizon pathway. This allows the denoising network to attend to a high-capacity, specialized representation of the immediate past while still retaining the full temporal context. Each conditioning token receives two additive embeddings before entering the cross-attention blocks: (i) a *temporal* embedding, a fixed sinusoidal positional encoding indexed by the token's original timestep index; and (ii) a learned *range* embedding that tags each token as LONG or SHORT, letting the network distinguish the two pathways (a reserved NULL entry is used for the prepended diffusion-timestep token). All tokens are linearly projected to a shared width before being attended to at every residual block of the 1D U-Net denoiser via multi-head cross-attention. To prevent the policy from over-relying on the short-horizon pathway, we apply *short-range dropout* during training: for each sample in a minibatch, with probability $p_{\text{short_range_dropout}}$ all T_s short-range tokens of that sample are replaced by a single learned null token $\mathbf{z}_{\varnothing} \in \mathbb{R}^d$. This dropout is disabled at inference time. In our experiments we use $T_s = 2$ and $p_{\text{short_range_dropout}} = 0.3$.

Note that we use the *past-token-prediction* auxiliary training objective from (Chi et al., 2024; Torne et al., 2025), where the policy's predicted action sequence includes actions corresponding to every

observation in the observation sequence. This means $T_o - 1$ predicted actions are in the past. In this paper, we treat the prediction horizon T_p as referring to only the predicted actions in the future and fix this at 15 timesteps, but the actual sequence the policy learns to predict has length $T_p + T_o - 1$.

For `PushT-D`, following (Torne et al., 2025; Agarwal et al., 2026), we use the baseline FiLM-conditioned U-Net with frozen observation encoder initialized from a trained short-context ($T_o = 2$) policy. For `Kitchen`, we use just the baseline FiLM-conditioned U-Net (fully fine-tuned for each context length). These simpler architectures proved sufficient for these tasks and show that our results generalize across architectures. As presented in (Agarwal et al., 2026), naive context length scaling can be sufficient for U-Net architectures in moderate-to-high data regimes.

G.2 Double Encoder Method Analysis

In this section, we provide a brief comparison between long-context policies (following the cross-attention U-Net architecture from (Agarwal et al., 2026)) trained with and without the double encoder on the `FurnitureSimOneLeg` task. See Fig. 14 for the results.



Figure 14: `FurnitureSimOneLeg` success-horizon curves for $T_o \in \{2, 4, 8, 12, 16, 20\}$ for policies trained with double encoder cross attention U-Net and cross attention U-Net (no double encoder) architectures on 200 human teleop demonstrations. Both plots exhibit similar trends (shorter T_{exec} perform better as T_o increases) and comparable absolute performance, but the double encoder architecture exhibits much less variation in performance across execution horizons, producing smoother and more interpretable curves.

While both yield similar absolute performance and exhibit the same trend from Sec. 5, where longer context lengths improve relative performance at low execution horizons (further supporting that our results are architecture-agnostic), the policy with the double encoder appears to have much less performance variation between execution horizons. Although the mechanism underlying this effect is unclear, we hypothesize that the double encoder may be providing redundant representations of short-range observations, reducing brittleness through either pathway. For this reason, we use the double encoder in most of our long context experiments.

G.3 Context Length Ablations for `PushT-M`

While our main context-length ablations in Sec. 5 use `PushT-D`, for completeness, we also train and evaluate long-context policies in `PushT-M`. Results are shown in Fig. 15.

As in `PushT-D` (Fig. 7), increasing context length shifts performance toward shorter execution horizons. However, unlike in our other tasks, absolute performance in `PushT-M` degrades substantially as context length increases. We hypothesize that this degradation reflects causal confusion exacerbated by visually sparse rendering in the ManiSkill simulator used in `PushT-M` — the lighting is bright and there are few visual features, causing unrelated states to appear similar in the learned representation (we observe that in some rollouts, the policy appears to mistake the T's upper corners for its base). This additional state aliasing may exacerbate causal confusion when conditioning on longer histories. In all other tasks, we find that multiple long-context training methods readily work out of the box. Because long-context training is not the focus of this work and this degradation may be specific to `PushT-M`'s simulation rendering, we leave it as future work to investigate further.

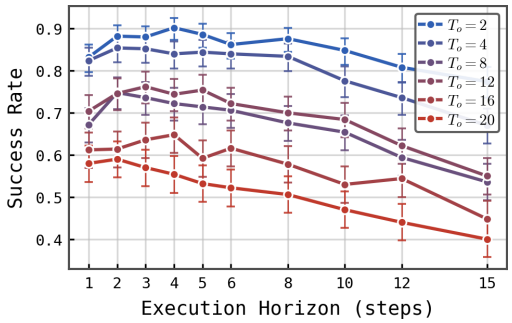


Figure 15: Success-horizon curves from long-context policies trained in `PushT-M`.

A separate observation is that the $T_o = 2$ curve in Fig. 15 can be compared directly with the human-expert `PushT` curve in Fig. 2. Both experiments use the same environment, demonstrations, and context length; only the architecture differs: the results here use the double encoder architecture, versus a FiLM-conditioned U-Net in Fig. 2. Both produce the same trend: an inverted U-shaped success-horizon curve with T^*_{exec} of 4 or 5. However, we notice an interesting phenomenon: the double encoder policy achieves substantially higher performance at $T_{\text{exec}} = 1$, suggesting architecture may also influence the success-horizon curve.

To explain this, we speculate that the double encoder's redundant representations and short-range dropout improve robustness to policy-induced states, thereby reducing compounding errors at low T_{exec}. Nevertheless, the improved architecture only changes the magnitude of the low-horizon performance deficit, not the qualitative preference: short-context policies imitating non-Markovian experts still prefer a non-trivial T_{exec} of 4 – 5. Meanwhile, increasing context length to improve observability of non-Markovian behaviors still produces a much more substantial shift of T^*_{exec} to 1 – 2. Thus, while architectural changes that reduce compounding errors may improve low-horizon performance, their influence on the success-horizon curve appears much weaker than changes in expert non-Markovianity.

H Additional Discussion: Multimodality and Smooth Execution

There is a subtle interplay between multimodality in expert demonstrations and smooth execution. Let us assume that every expert demonstration trajectory is smooth,[5] but over the batch of demonstrations there are cases where the same history can lead to qualitatively different actions (multimodal), e.g. as might have been the case for Buridan's Ass. Since the learned policy mimics the *distribution* of actions, the concept of smoothness coming from each individual trajectory might be lost, and one might worry again about the learned policy cycling between two different "modes" in the demonstration data.

Let us assume for the moment that the learner learns perfectly (reconstructing the true distribution optimally given the available context). If the history/context of the learner is the same or greater than the expert, then all sampled trajectories must also be smooth (non-smooth trajectories would have zero probability mass). The more interesting case is where the learner has a more limited context than the expert. Smoothness is only a local property of the trajectories and does not require arbitrarily long histories. Technically, we are working with sampled-data systems here, so we can only talk about "discrete smoothness" (e.g., high-order differences quickly decay); if the history contains k recent samples at the execution sampling rate, then we expect that the outputs will be discretely smooth to order $k - 1$.

Thus, under perfect learning, we argue that expert multimodality should not by itself cause arbitrarily unsmooth execution. Learning errors (including due to finite demonstration data) can complicate this, but we leave that analysis to future work.

[5]Here, "smoothness" refers qualitatively to discrete-time smoothness of the action sequence, i.e., actions and their higher-order discrete derivatives vary gradually over time.

I Additional Discussion: Benefits of Reactivity in Our Experiments

In Sec. 4.1.2, we find that **reactivity is consistently beneficial even in quasi-static tasks:** We note that none of our tasks are inherently dynamic or unpredictable. Nevertheless, across all tasks, policies trained on Markovian expert data perform best at or near $T_{\text{exec}} = 1$, when they are most reactive. These experiments show that, even in seemingly static tasks, errors in execution and in the learned dynamics are nearly unavoidable, making reactive feedback broadly beneficial in manipulation, not only in inherently dynamic settings. While ablating T_{exec} is not common in prior work, selecting the best T_{exec} can yield substantial improvements of tens of percentage points in absolute success rates.

We now provide more concrete examples of the benefits of reactivity in our experiments.

We primarily attribute its benefits to challenging contact dynamics (particularly in `FurnitureSimOneLeg`, during the screwing phase, and in `Push-T`) and high precision requirements (particularly in `FurnitureSimOneLeg` and `GearInsertion`). In `FurnitureSimOneLeg`, at high T_{exec}, we find a vast majority of failures occur during insertion (by missing the socket) and screwing (due to the table top sliding away); these failures become very infrequent at low T_{exec} where the policy can re-aim and adjust when the table begins to slide. Most high-T_{exec} failures in `GearInsertion` occur due to missed insertion; in `Push-T` — due to unexpected T motion that require additional pushing maneuvers (leading to timeout); in `Kitchen` — due to lack of precision when grasping objects.

These failure modes illustrate that even nominally static manipulation tasks suffer inevitable execution errors, from imperfect policy learning, control, interaction dynamics, or otherwise. Reactivity allows policies to quickly correct such errors, providing beneficial robustness even in otherwise static settings.

J Additional Discussion: The Importance of Checkpoint Selection for Long-Context, Reactive Policies

Across almost all experiments, we noticed two consistent trends. While conventional Diffusion Policy wisdom promotes over-training past the epoch of minimum validation loss (He et al., 2025) to achieve strong rollout performance, we find:

1. **Lower execution horizons prefer earlier checkpoints, i.e. less over-training.**
2. **Longer context lengths also prefer earlier checkpoints.**

Claim 1 holds fairly consistently across all experiments. We find claim 2 less consistent — we observe it strongly across the `FurnitureSimOneLeg`, `SinglePillDispense`, and `SlipIntoBaggie` experiments; in `SinglePillDispense`, naively evaluating the latest checkpoint even yields up to 40% lower success rates compared to an earlier checkpoint. But we observe no pattern at all in `Push-T`, `GearInsertion`, and `Kitchen`. We also acknowledge that these comparisons conflate other variables (i.e. network architecture), but nonetheless motivate evaluating checkpoints throughout training — a precaution that may be crucial in some settings but less so in others.

We plot the optimal training duration against execution horizon in Fig. 16, and against context length in Fig. 17.

One possible explanation for this phenomenon is that early-stopping training reduces overfitting and improves generalization to unseen states (Raskutti et al., 2013), which is especially important when using long context lengths or short execution horizons — long context lengths expand the policy's input space, increasing risk of encountering out-of-distribution inputs during rollout (see Sec. 5), while short execution horizons allow errors on policy-induced out-of-distribution states to compound more readily (see Sec. 4.2). **The implication is that, to effectively train policies with either long context lengths and/or short execution horizons, adopting careful checkpoint selection processes is critical.**

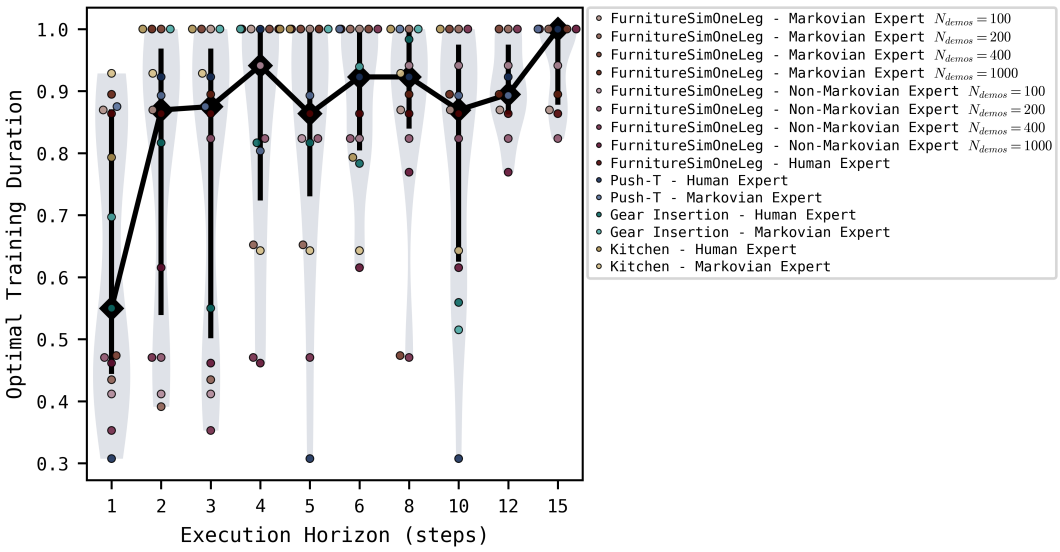


Figure 16: Distribution of the optimal training duration (calculated as the fraction of the highest-performing checkpoint's epoch divided by the latest evaluated checkpoint's epoch) as a function of execution horizon, pooled across all short-context ($T_o = 2$) scripted- and human-expert experiments (individual points represent individual experiments; black diamonds/bars represent median and interquartile range). We observe a consistent trend across almost all experiments that lower execution horizons prefer earlier checkpoints.

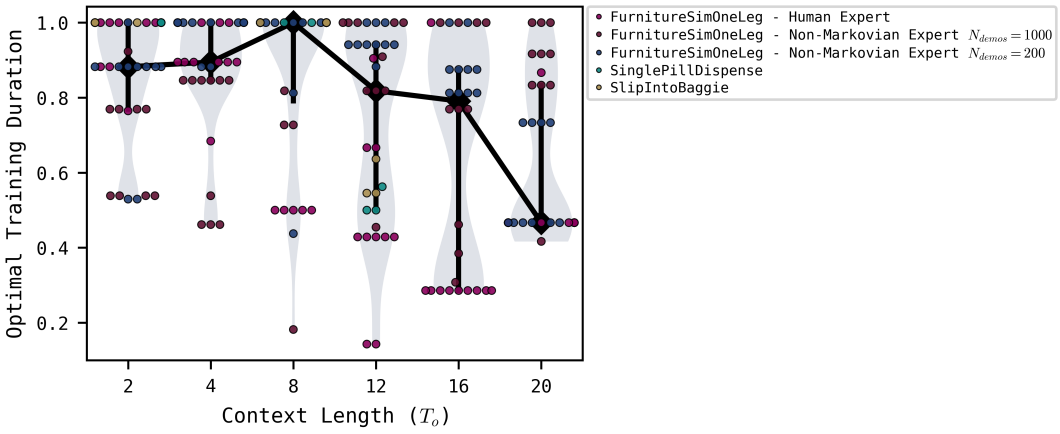


Figure 17: Distribution of the optimal training duration as a function of context length for all `FurnitureSimOneLeg`, `SinglePillDispense`, and `SlipIntoBaggie` context length experiments (including all execution horizons). We observe policies with longer context lengths often perform better at earlier training checkpoints.

This implication may become even more important at large data scales. Results from Generalist's GEN-0 and Sunday's ACT-2 suggest that, with enough data, validation loss becomes a strong predictor of closed-loop performance (Generalist Team, 2025; Sunday Robotics, 2026). In this regime, policies no longer face the same tradeoff between generalization and rollout performance: they can generalize well *and* perform strongly without over-training. We therefore hypothesize that long-context, reactive policies will become even more favorable at scale. Nevertheless, we note that this analysis is preliminary, and much remains to be understood about the relationship between checkpoint selection, validation loss, and rollout performance in behavior cloning.

In relation to Sec. 4.2, this result further contributes that compounding errors especially impact lower execution horizons; but proper checkpoint selection is a simple and standard way to limit this effect.