REQUERY IS NOT FREE: CROSS-QUERY COUPLING IN GENERATIVE ROBOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Action-chunked policies have become a widely adopted and effective approach to robot control. These policies repeatedly generate new action chunks from fresh observations, allowing them to adapt during execution. Yet each new chunk must continue a trajectory shaped by earlier actions. Fresh observations update what the policy sees, but may not reveal how that trajectory was reached. A new chunk may therefore fit the current observation yet be incompatible with the execution already underway. We formalize this dependence as *cross-query coupling* and prove that identical per-query chunk distributions can yield exponentially different trajectory success. In learned policies, we find that a chunk's value depends on the physical trajectory it continues, with particularly pronounced effects in some tasks and execution stages. This diagnosis motivates STATEFUL REQUERY, which combines fresh observations with a compact record of realized execution. The record preserves what earlier actions actually changed, giving each new generation context for continuing the current trajectory. The method improves control across all tested backbone–generator combinations on Meta-World MT50 and LIBERO-90. Analysis shows how the record guides branch-specific action adjustments. It remains useful even when the policy requeries after every action. These findings identify requery as a trajectory-composition problem, where effective control requires action chunks that are not merely plausible but appropriate for the execution they continue.

1 INTRODUCTION

Modern generative robot policies produce action chunks, execute a prefix, and use fresh observations to generate the next chunk (Zhao et al., 2023; Chi et al., 2023). We call this repeated generative handoff *requery*. It lets the robot respond to feedback rather than finish an obsolete plan. But generating anew does not mean starting over physically. Earlier actions have created a physical branch that the next chunk must continue even when it replaces the previous plan.

The input to a query may not distinguish histories that leave the robot in different physical situations. Per-query imitation then learns from their combined continuations, even though an action that advances one branch may work against another. A better fit to this mixture can improve the plausibility of individual chunks without identifying which continuation suits the branch reached by earlier execution. Requery therefore requires both generating plausible candidates and matching them to the physical trajectory already underway (Figure 1).

Providing more history is a natural response, yet Diffusion Policy's ablations favor short visual windows over longer ones (Chi et al., 2025). In VLANeXt's LIBERO Spatial ablation, adding past frames lowers success from 91.8% to 85.0% (Wu et al., 2026). Methods that structure the context carried between decisions can instead produce large closed-loop gains (Lian et al., 2026; Liang et al., 2026b; Huang et al., 2026; Jiang et al., 2026). The question is therefore what the next query needs to retain about execution, not simply how far into the past it should look.

We formalize the pairing between an execution-created branch and its next chunk as *cross-query coupling*. We prove that policies with identical per-query chunk distributions can have exponentially different trajectory success. What differs is how those chunks continue earlier execution. We also characterize which historical distinctions change the relative value of continuations. This criterion specifies what information an interface must retain to preserve those value differences across queries:

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.

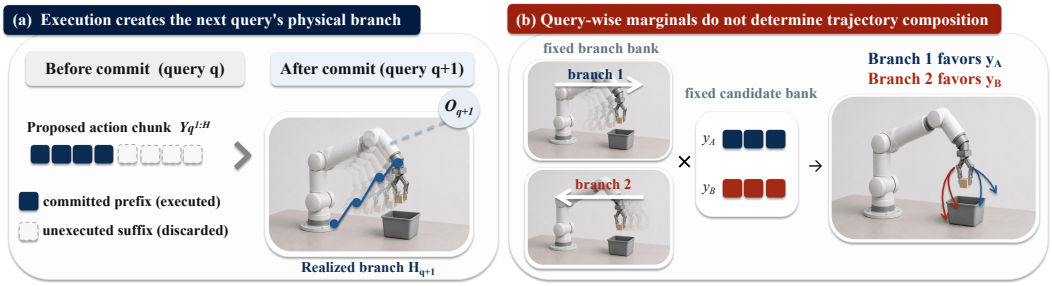


Figure 1: **Requery couples successive generations through execution.** **(a)** Only the committed prefix becomes physical and creates the branch faced by the next query. **(b)** With the branch and candidate banks fixed, branch 1 favors y_A, whereas branch 2 favors y_B. A chunk's suitability therefore depends on the physical branch it continues.

distinctions matter when they change which continuation is useful, not merely when they reveal more about the past.

In learned Flow and Diffusion policies, candidate rankings vary widely across physical branches, with larger value differences in particular tasks and execution stages. We establish this dependence by executing the same candidate chunks on different receiving branches, directly testing the criterion above. Along natural rollouts, states with closely matched observations also favor different continuations from the same candidate bank. The controlled test isolates dependence on the receiving branch; the natural comparison locates it in physical situations the policies actually reach. Continuation quality thus depends jointly on the generated chunk and the physical branch receiving it.

The diagnosis suggests preserving execution context while refreshing perception. Commands specify what the robot was asked to do; realized transitions record how it responded. Retaining these responses tells the next query how earlier actions shaped the current branch. STATEFUL REQUERY combines this compact record with fresh observations when generating each new chunk. The record is updated by what the robot actually executes, rather than by the unused remainder of a plan. Across complete Transformer/U-Net and Flow/Diffusion crosses on Meta-World MT50 and LIBERO-90, the method improves mean success across three independent training seeds in every model–benchmark cell.

The benefit of this record helps clarify what history contributes to requery. Our analysis shows that information beyond the latest physical response remains useful even as fresh observations arrive. The policy can use this information to adjust its actions in ways that improve continuation on the receiving branch. Execution history thus complements current perception by preserving consequences of earlier actions that still matter for what the robot should do next.

Contributions. Our contributions are threefold:

- **A decision-theoretic foundation for cross-query coupling.** We show that identical per-query chunk distributions can yield exponentially different trajectory success. We characterize exactly which history distinctions a requery interface must preserve to maintain relative continuation values, and connect branch–chunk pairing to trajectory return across queries.
- **An executable diagnosis of branch-relative continuation value.** We turn the criterion into a fixed-bank test, exposing branch-dependent value in controlled interventions and observation-matched natural trajectories. Task and stage analyses locate where this dependence has the greatest consequence.
- **A commitment-gated interface with trajectory-level validation.** We separate fresh evidence from committed execution and instantiate the resulting principle with STATEFUL REQUERY. Trained-policy comparisons establish its control benefit. Changing the record read at a handoff reveals branch-specific action adjustments and consequences that extend through later queries.

2 PER-QUERY MARGINALS DO NOT DETERMINE TRAJECTORY COMPOSITION

A chunk distribution specifies which actions a query can generate, but not how those actions are paired with the physical branches created by earlier execution. We ask when this pairing changes value, what information makes it irrelevant, and how its consequences propagate through a rollout.

2.1 FROM PER-QUERY MARGINALS TO AN EXECUTION-COUPLED LAW

Let H_q be the complete realized history before query q, and let $I_q = \phi(H_q)$ be the finite interface exposed to the generator. One query, physical commitment, and return to the interface obey

$$\xi_q \sim p_\xi, \quad Y_q = G_\theta(I_q; \xi_q), \qquad H_{q+1} \sim \mathcal{T}(\cdot \mid H_q, Y_q^{1:E}), \quad I_{q+1} = \phi(H_{q+1}).$$

Only the committed prefix $Y_q^{1:E}$ becomes physical. Its execution changes the next query's history, linking successive calls through the trajectory even when each draws fresh noise.

Histories giving the same input i are indistinguishable to the generator; they form the interface fiber $\mathcal{F}_i = \{h : \phi(h) = i\}$. At population optimum, query-wise supervision mixes their expert continuations $K^*(dy \mid i, h)$ over the training distribution $\mu_i^{\mathcal{D}}$. At deployment, let μ_i^π be the history distribution reached by policy π and ν_i^θ the generator's chunk law:

$$\nu_i^{\mathcal{D}}(dy) = \int_{\mathcal{F}_i} K^*(dy \mid i, h)\, \mu_i^{\mathcal{D}}(dh), \qquad \gamma_i^{\mathrm{prod}}(dh, dy) = \mu_i^\pi(dh)\nu_i^\theta(dy). \tag{1}$$

With fresh noise independent of history given i, the product law gives each branch the same chunk distribution. Earlier execution shapes μ_i^π, which may differ from training. Matching the training mixture leaves open how samples fit the branches reached.

To isolate this issue, hold fixed how often each branch and each chunk appears, and change only which branch receives which chunk. Write these marginals as μ_i, ν_i and their possible joint laws as $\gamma_i \in \Gamma(\mu_i, \nu_i)$. We call γ_i the *cross-query coupling*. A refined interface can condition generation on the branch while preserving both marginals.

The distinction can be exponential. If each of Q queries exposes K equiprobable modes and success requires one mode to persist, independent resampling succeeds with probability K^{1-Q}; retaining one uniform mode succeeds surely. Every per-query marginal is identical. The missing object is therefore how choices compose: equal mode frequencies can accompany very different trajectories.

2.2 WHICH HISTORY DISTINCTIONS MATTER FOR CONTINUATION?

When do histories sharing an input change one candidate's advantage over another? For a task, downstream policy π, and n remaining calls, let $Q_n^\pi(h, y)$ be the expected return after executing y's committed prefix at h. Relative to a reference candidate y_0, define

$$\rho_{i,n}(h, y) = Q_n^\pi(h, y) - Q_n^\pi(h, y_0).$$

An interface is *n-continuation-sufficient* when this candidate-relative profile is constant over $h \in \mathcal{F}_i$. A rank switch certifies failure, but even a common winner is insufficient: return pairs $(2, 0)$ and $(4, 3)$ both favor the first candidate while assigning different advantages to it. The following criterion explains why these value gaps determine whether pairing matters.

On finite supports where every cross-pair is executable, define

$$J_{i,n}(\gamma) = \mathbb{E}_{(H,Y)\sim\gamma}[Q_n^\pi(H, Y)]. \tag{2}$$

Exact fixed-marginal null. $J_{i,n}(\gamma)$ is identical for every $\gamma \in \Gamma(\mu_i, \nu_i)$ if and only if

$$Q_n^\pi(h, y) = a_i(h) + b_i(y) \quad \text{on } \mathrm{supp}(\mu_i) \times \mathrm{supp}(\nu_i). \tag{3}$$

Equivalently, every candidate gap is unchanged across histories in the fiber:

$$Q_n^\pi(h, y) - Q_n^\pi(h, y') = Q_n^\pi(h', y) - Q_n^\pi(h', y'), \qquad h, h' \in \mathcal{F}_i.$$

History then shifts all candidates' values by the same offset $a_i(h)$, preserving their relative merits. Otherwise, some exchange of two candidates between two histories changes value while preserving both marginals. This 2×2 argument proves the criterion (Appendix A.1).

The criterion also identifies what the interface must retain. For at most N remaining queries, group histories only when their relative profiles agree for every $n = 1, \ldots, N$. This *continuation quotient* is the coarsest supported partition preserving candidate-relative value through the horizon. It retains distinctions that affect continuation, rather than reconstructing the entire history or latent state.

When the null fails, the range of values attainable by re-pairing quantifies the remaining ambiguity. With $\Gamma_i = \Gamma(\mu_i, \nu_i)$, define the *coupling diameter*

$$\mathfrak{C}_{i,n}(\mu_i, \nu_i) \coloneqq \max_{\gamma \in \Gamma_i} J_{i,n}(\gamma) - \min_{\gamma \in \Gamma_i} J_{i,n}(\gamma). \tag{4}$$

This diameter is zero exactly at the fixed-marginal null. Approximate additivity bounds its magnitude: with $\|\cdot\|_{\infty,i}$ over the supported cross-product,

$$\varepsilon_{i,n} = \inf_{a,b} \left\| Q_n^\pi - a - b \right\|_{\infty,i}, \qquad \mathfrak{C}_{i,n}(\mu_i, \nu_i) \le 2\varepsilon_{i,n}. \tag{5}$$

For an executable diagnostic, ask whether knowing the branch improves the choice from a finite candidate bank B:

$$G_{i,n}(B) = \mathbb{E}_H \left[\max_{y \in B} Q_n^\pi(H, y) \right] - \max_{y \in B} \mathbb{E}_H[Q_n^\pi(H, y)] \ge 0. \tag{6}$$

$G_{i,n}(B)$ compares the best choice for each branch with one choice for all branches. It measures the value of branch-aware selection from the same bank, allowing candidate frequencies to change. A positive gap witnesses failure of continuation sufficiency; zero is inconclusive because a common winner can conceal changing value gaps.

2.3 From one pairing to trajectory value

Histories can agree on candidate values for the next segment yet disagree on which continuation becomes useful afterward. Each commitment creates a new handoff, so preserving the current ranking alone does not resolve future decisions. One sufficient solution is a statistic that determines each candidate's expected reward and distribution over the next statistic. This information-state closure preserves continuation value across queries (Subramanian et al., 2022).

To connect these decisions to rollout return, let $(\mathcal{B}_q^A V)(h) = \mathbb{E}_A[R_q + V(H_{q+1}) \mid H_q = h]$ denote the query-q Bellman operator for mechanism A, with segment reward R_q. It includes generation, execution, dynamics and interface update. Write d_q^A for its reached-handoff distribution and V_{q+1}^B for B's remaining value. For mechanisms sharing an initial distribution, the performance-difference identity (Kakade & Langford, 2002) gives

$$J^A - J^B = \sum_{q=1}^{Q} \mathbb{E}_{H_q \sim d_q^A} \left[\left((\mathcal{B}_q^A - \mathcal{B}_q^B) V_{q+1}^B \right)(H_q) \right]. \tag{7}$$

Each local difference is evaluated on the handoffs created by all earlier queries. Within each fiber reached under d_q^A, compare both mechanisms on that same history distribution. If their candidate marginals also agree, the fiber-averaged term is a difference of the coupled values in equation 2. Hence $|J^A - J^B|$ is bounded by $\sum_q \mathbb{E}_{H_q \sim d_q^A}[\mathfrak{C}_{\phi(H_q), Q-q+1}]$; applying equation 5 replaces each diameter by $2\varepsilon_{\phi(H_q), Q-q+1}$. Continuation sufficiency makes every marginal-preserving pairing term vanish. Without it, an earlier pairing can change which high-diameter handoffs later queries occupy, so a negligible one-query contrast need not remain negligible.

The empirical question is which branch-dependent distinctions matter in learned policies, and how execution history helps the generator act on them. We first examine how the same candidates change in value across physical branches, then study how retaining realized execution shapes continuation across successive queries.

3 Continuation Value Is Branch-Conditioned in Learned Policies

Section 2 asks whether a chunk's advantage over another changes with the physical branch. We test this dependence across tasks, execution stages and natural rollouts.

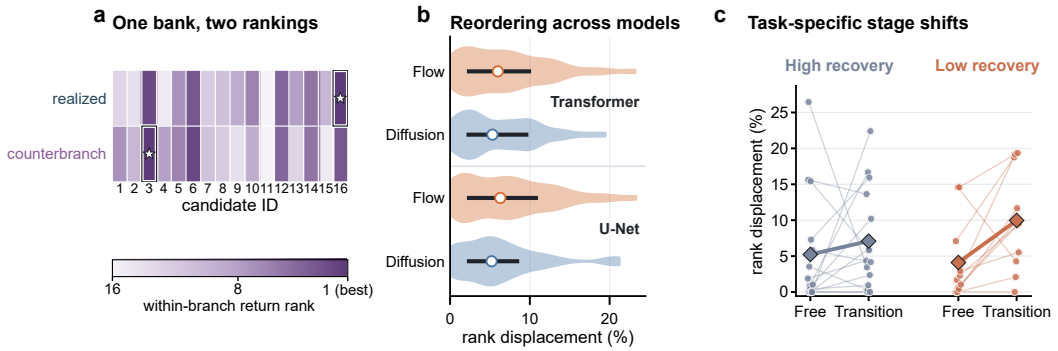


Figure 2: **Branch-dependent rankings vary across tasks and execution stages.** **(a)** One fixed 16-candidate bank has different winners on two restored branches (c_{16} versus c_3). **(b)** Candidate-rank changes across 50 tasks for each of the four models; markers show medians and interquartile ranges. **(c)** Free-motion versus pooled contact-onset and release-transition means for 15 tasks with more room for correction and 11 with less (high/low recovery), averaged over Transformer–Flow and U-Net–Diffusion at shared expert query boundaries. Lines join tasks; diamonds show group means.

3.1 A FIXED BANK EXPOSES BRANCH-CONDITIONED VALUE

We generate 16 candidate chunks from one two-frame input and execute this fixed bank from two states: the original state and a counterbranch with the same positions but reversed velocities. The next 16 actions' cumulative return measures branch-dependent immediate progress from the same proposals; original-state repeats measure replay variation. For Transformer–Flow and U-Net–Diffusion, the study spans 50 Meta-World tasks and 79,200 short executions at query boundaries from policy and expert trajectories. A shared set of 400 expert boundaries covers all four Transformer/U-Net $\times$ Flow/Diffusion combinations.

Rank divergence measures how far candidates move up or down the ranking, normalized by bank size. The value gap G_{excess} measures the advantage of choosing the best candidate for each branch over choosing one for both, after return normalization and replay correction. A rank switch violates equation 3; a positive gap witnesses the branch-aware selection advantage in equation 6.

Changing the branch changes which chunks work best. For both original models, the empirical coupling diameter is positive in 48–49 of 50 tasks on both policy and expert trajectories. Across all four models, the best candidate switches at 18–21% of shared query boundaries. Across Flow, Diffusion and deterministic action chunking, task means favor a fresh chunk about as often as the unexecuted suffix of the previous chunk. Fresh generation alone does not resolve which continuation fits the branch.

3.2 BRANCH SENSITIVITY VARIES WITH TASK GEOMETRY AND EXECUTION STAGE

The four models broadly agree on which tasks show the strongest rank shifts (Spearman $\rho = .707$–$.855$; Figure 2b). In each model, ten tasks account for 78–91% of the positive value gap. Task geometry helps explain this concentration. Insertion, pressing, locking, constrained placement and tool contact leave less room for correction than reaching, pushing or sweeping. We use these semantics, not measured returns, to define low- and high-recovery groups. Low-recovery tasks show 44% greater mean rank divergence and a 72% larger value gap.

Within a task, we compare free motion with a pooled transition group covering contact onset and release, identified from preceding contact, object motion and gripper motion. This comparison tracks changes in interaction within the same task rather than differences between tasks. On shared expert trajectories, low-recovery tasks have lower mean rank displacement during free motion but higher displacement at transitions (Figure 2c). Their free-to-transition increase exceeds that of high-recovery tasks by 4.00 points; the corresponding difference in G_{excess} is 1.11×10^{-2}. Despite substantial task variation, the pattern locates where branch-compatible choice becomes more consequential: not simply in particular tasks, but when their interaction conditions change.

3.3 BRANCH DEPENDENCE ALSO APPEARS IN NATURAL EXECUTION

Natural policy trajectories also contain states that look similar but favor different continuations. We pair query boundaries with closely matched images and proprioceptive observations, then execute the same candidate bank from both states. Across 1,600 natural trajectories and 84,480 short executions, the best candidate changes in 28.8% of comparisons for Transformer–Flow and 22.6% for U-Net–Diffusion, averaging equally over the 49 and 41 tasks with matched branches, respectively. The effect persists when the allowed differences in both images and proprioception are halved. Branch-dependent preferences thus also occur at query boundaries the policies naturally reach.

4 STATEFUL REQUERY SYNCHRONIZES GENERATION WITH COMMITTED EXECUTION

The branch-dependent preferences above motivate giving each query information about how its current physical situation arose. Realized transitions record how the robot responded to earlier actions, helping distinguish histories that fresh perception may merge. STATEFUL REQUERY carries this record alongside fresh observations. The synchronization point is physical commitment: state is written after a prefix executes and before the next generation. Unexecuted plans are excluded.

4.1 A COMMITMENT-GATED SYNCHRONIZATION POINT

Let O_q denote the fresh perceptual and proprioceptive input, and m_q a fixed-capacity execution state, initialized to zero each episode. Stateful Requery refines the interface to $I_q^{\mathrm{SR}} = (O_q, m_q)$:

$$Y_q \sim \pi_\theta^{\mathcal{G}}(\cdot \mid O_q, m_q), \qquad m_{q+1} = U_\phi(m_q, \mathcal{E}_q), \tag{8}$$

where $\mathcal{E}_q$ contains only transitions realized by executing the committed prefix $Y_q^{1:E}$. Each event records the pre-transition proprioceptive state, realized motion, and elapsed time. Thus the writer sees what the robot did, not just what the generator asked it to do. The generator reads this accumulated record to condition the next chunk.

At each query (Figure 3), we freeze m_q, generate a chunk, execute its prefix, write realized events, and discard the unused suffix. Generator internals and unused plans never enter persistent state. Perception can revise action immediately; persistence records only committed physical events.

4.2 ONE FIXED-CAPACITY REALIZATION

We realize the contract with Kimi Delta Attention (KDA) (Kimi Team, 2025). Its fixed-capacity associative matrix supports content-addressed reads, while its delta update can revise retained content rather than merely append to it. Each realized transition writes a gated update; the generator reads the resulting state through a gated residual. KDA is useful here because commitments must persist without becoming immutable. Full equations and deployment pseudocode appear in the appendix.

Transformer action tokens read the persistent matrix; U-Net reads through the global observation condition. The backbone fixes this path; Flow and Diffusion set the objective and sampler. Learned retention and read gates decide which earlier events remain useful as the rollout grows.

4.3 LEARNING THE DEPLOYED RECURRENCE

Training follows the same read–execute–write order. Before query q, the writer has seen only realized transitions preceding its target. We optimize

$$\min_{\theta,\phi} \; \mathbb{E}_{\tau\sim\mathcal{D}} \left[\sum_{q=1}^{4} \ell_{\mathcal{G}}\big(\theta; Y_q^\star, O_q, m_q\big) \right], \qquad m_{q+1} = U_\phi(m_q, \mathcal{E}_q^\star), \tag{9}$$

with the native flow-matching or diffusion objective unchanged. During training, earlier demonstrated transitions update the state; at deployment, sampled and realized transitions do. Unexecuted suffixes never write. The loss remains query-wise; chronology changes the next query's input.

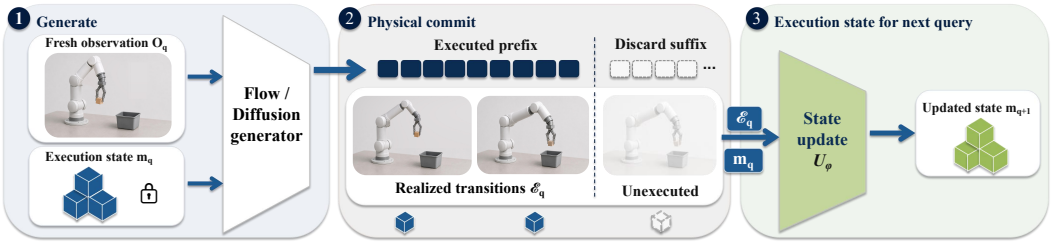


Figure 3: **Stateful Requery is commitment-gated.** The execution state is read-only during generation; only transitions produced by the committed prefix update m_{q+1}.

Gradients from later queries teach the writer which earlier transitions help predict subsequent actions. Training therefore rewards retention through its contribution to those predictions, giving the record a role in subsequent decisions without requiring it to reconstruct the entire execution history.

5 Stateful Requery: Execution History Shapes Continuation and Improves Control

We first establish Stateful Requery's control benefit, then examine what its execution record contributes through fresh feedback and how it shapes branch-appropriate actions.

5.1 The interface improves full trajectories

On the full 50-task Meta-World and 90-task LIBERO benchmarks, Stateful Requery improves three-seed mean success in every structural pairing (Table 1). The benefit spans both backbones and both generators, including policies with strong observation-only performance.

Table 1: **Stateful Requery transfers across complete structural crosses.** Three-seed mean task-balanced success (%); parentheses give within-cell gains (percentage points).

		Flow		Diffusion	
Benchmark	Backbone	Observation	Stateful	Observation	Stateful
Meta-World MT50	Transformer	73.25	**77.96** (+4.71)	50.49	**51.72** (+1.23)
	U-Net	79.07	**79.99** (+0.92)	65.21	**70.89** (+5.68)
LIBERO-90	Transformer	24.34	**67.64** (+43.30)	53.21	**69.95** (+16.74)
	U-Net	36.93	**80.89** (+43.96)	68.14	**83.27** (+15.13)

The gains are larger on LIBERO-90, where policies control seven-dimensional actions rather than the four-dimensional actions in Meta-World. Stateful Requery remains effective in this higher-dimensional control setting, with both Flow backbones gaining more than 43 percentage points.

5.2 Committed transitions carry the gain

Commands record what the policy asked the robot to do; realized transitions record how it responded. Table 2 compares what the requery interface retains. Command records retain either the latest executed prefix or commands accumulated across the episode. Realized-transition records accumulate pre-transition states and observed motion. The no-time variant omits physical elapsed time; Stateful Requery uses the full record defined in Section 4.

Realized transitions improve over observation-only conditioning in all four models, whereas both command records reduce success. The useful distinction is between a requested motion and the robot's response: the same command can produce free motion or encounter contact, leaving different physical situations for the next chunk to continue.

Table 2: **Execution-grounded carriers improve control across model families.** Within-family gain over each column's observation-only policy on Meta-World MT50 (percentage points), using the canonical seed-42 models.

	Transformer		U-Net	
Interface content	Flow	Diffusion	Flow	Diffusion
Latest executed commands	-18.24	-14.60	-20.96	-14.92
Accumulated executed commands	-15.08	-7.32	-20.24	-15.40
Realized transitions; no elapsed time	+7.64	+0.84	+1.92	+7.80
Stateful Requery (full record)	+4.48	+2.12	+1.28	+6.16

Restricting the record to the latest executed prefix reduces Transformer–Flow success from 76.44% to 58.52% on MT50. This *latest-prefix* control retains only the newest realized transitions; both designs are trained with their respective history scope, using the same two-frame observations and writer/read path. Earlier execution therefore provides useful context beyond the robot's latest response.

Retaining execution also differs from retaining predictions. Temporal ensembling averages overlapping action predictions across queries (Zhao et al., 2023), whereas Stateful Requery uses realized transitions to condition generation. Applying the same averaging to both policies leaves Stateful ahead by 2.36 points in Transformer–Flow and 5.72 points in U-Net–Diffusion. Execution conditioning therefore remains useful alongside prediction aggregation.

5.3 Execution context remains useful through fresh feedback

Fresh feedback does not replace the accumulated execution record. At the standard execution interval, all four Stateful policies perform better when reading their record than when presented with zero state at every query. This dependence remains when feedback arrives after every action. On the 24-task horizon comparison, retaining state rather than resetting it raises success by 42.19 points in Transformer–Flow and 66.41 points in U-Net–Diffusion at $E = 1$. With no multi-action interval between observations, the record still provides useful context for generation.

This benefit extends across several queries. At fixed $E = 16$, all four backbone–generator combinations improve as the interval between execution-state resets grows from one to two to four queries, then to the full episode. The record must also incorporate new execution. In an eight-task Flow control, omitting the most recently executed prefix from the record reduces next-segment progress. Retention and updating thus preserve complementary parts of the execution context.

5.4 Execution history realizes branch-specific repair

Does correct history improve the robot's next continuation, and is that improvement specific to its physical branch? We pair handoffs from the same task and stage, each with history accumulated from episode start. At each receiver, its own history and its partner's generate two candidate banks under fixed observations, weights and noise. We execute each bank on the receiver $(+)$ and on its velocity-mirrored counterbranch $(-)$, crossing history-conditioned actions with the physical branches they must continue.

Let D_b be the difference in mean bank return between own and partner history on branch b. Then $\Delta_{\text{native}} = D_+$ measures the receiver's benefit, while $\Delta_{\text{align}} = D_+ - D_-$ measures how much that benefit favors its branch. Improving both branches equally would raise the first contrast but leave the second unchanged. Positive values of both establish useful branch-specific repair. This is Section 2's rectangular interaction: with equal use of both banks and branches, diagonal pairing gains $\Delta_{\text{align}}/2$ over crossed pairing.

A reciprocal pick–place pair demonstrates both effects in Transformer–Flow and U-Net–Diffusion. Both histories generate forward and upward motion that lifts the object. Matching history preserves the lift but adjusts lateral motion in opposite directions at the two receivers. Each receiver's own history yields higher return across all paired draws. Reversing receiver roles reverses the useful adjustment, showing how execution history adapts a shared movement to the branch being continued. Crossing the banks over physical branches verifies that the value advantage depends on this pairing.

Each action also changes the state from which later queries continue. A separate recursive comparison follows this dependence under fresh feedback, starting two rollouts from the same physical handoff, one reading its own history and the other initially reading history from another rollout of the same task. After this single swap, both receive fresh observations and write their own realized transitions. From one to six future queries, the own-minus-swapped progress gap, measured by reduction in object–goal distance, grows from 0.269 to 1.225 mm for Transformer–Flow and from 0.174 to 2.216 mm for U-Net–Flow. The two Diffusion cells do not resolve this temporal separation.

The performance identity in equation 7 connects these scales: each local decision is evaluated on handoffs created by its predecessors. History-dependent repair thus concerns both the current action and the setting of later decisions.

6 Related Work

Action chunking and continuation. ACT and Diffusion Policy established multi-step closed-loop prediction (Zhao et al., 2023; Chi et al., 2023). Subsequent analyses examine stability, imitation-error propagation (Simchowitz et al., 2025; Zhang et al., 2026) and iterative generation (Pan et al., 2026). Candidate selection (Liu et al., 2025), guided continuation (Liu et al., 2026), plan reuse (Black et al., 2025; Zhan et al., 2026) and adaptive querying (Liang et al., 2026a; Nie et al., 2026) address consistency and reactivity. We study the complementary problem of pairing a generated chunk with the physical trajectory it must continue.

Decision abstraction under partial observability. Belief and predictive states (Kaelbling et al., 1998; Singh et al., 2004), recurrent policies (Hausknecht & Stone, 2015), and value abstraction (Abel et al., 2020; Misra et al., 2020) address how history supports decisions. Our formulation specializes this question to requery. Histories matter when they change the relative value of possible continuations, even if the current observation looks the same. Continuation sufficiency identifies which such distinctions an interface must retain across successive decisions.

Execution context across queries. IntentVLA addresses aliasing and intent switching with recent visual history (Lian et al., 2026). TFP learns event-sensitive memory dynamics and tests their influence through hidden-state interventions (Liang et al., 2026b). ChainVLA combines accumulated execution context with the previous prediction's unexecuted tail (Huang et al., 2026). Fast weights, tokens, and event memories offer further carriers (Jiang et al., 2026; Jang et al., 2025; Zeng et al., 2026). These approaches carry context across queries. We ask what execution information makes a continuation appropriate for its physical branch. Stateful Requery retains the robot's realized responses to earlier actions.

7 Discussion and Conclusion

Requery asks how to continue a physical trajectory, not simply when to observe again. Continuation sufficiency makes this dependence precise: an interface must preserve history distinctions that change candidate-relative value through subsequent queries.

Stateful Requery provides a persistent channel for those distinctions alongside fresh perception. In the reciprocal repair, history preserves a shared lift but changes the lateral adjustment needed by each receiver. The design question is which consequences of execution to retain, and when new evidence should revise their influence.

Our experiments use simulated manipulation, leaving robustness to real sensing and contact uncertainty untested. The current record is trained through action imitation. Supervising it with branch-relative continuation values could teach which past transitions change the useful continuation and which can be compressed away, complementing the single action sequence provided by each demonstration. This would turn continuation sufficiency from a diagnostic criterion into a target for learning.

The requery interface is part of the policy itself. Fresh observations update the evidence but do not restart the world. A generative controller must use that evidence to continue the physical trajectory created by its earlier actions.

Reproducibility statement

The appendix provides complete proofs, implementation details, and experimental protocols. Code, experiment configurations, and analysis scripts will be made publicly available.

Ethics statement

This study uses simulated robot-manipulation environments and existing benchmark datasets, without conducting human-subject experiments or collecting personal data. Its findings concern control performance rather than safety certification. Deployment on physical robots would require separate safety validation and safeguards against unintended actions.

AI use statement

Generative AI tools were used to assist with mathematical verification, implementation support, and manuscript editing. The authors take full responsibility for the final text, claims, code, and results.

References

David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1639–1650, 2020. URL https://proceedings.mlr.press/v108/abel20a.html.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.07339.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

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. *The International Journal of Robotics Research*, 2025. doi: 10.1177/02783649241273668. URL https://diffusion-policy.cs.columbia.edu/diffusion_policy_ijrr.pdf.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015. URL https://arxiv.org/abs/1507.06527.

Yuzhi Huang, Weijue Bu, Ziyi Xiong, Jie Wu, Fanding Huang, Jingyan Jiang, and Zhi Wang. ChainVLA: Chaining vision-language-action queries through a unified execution state for long-horizon manipulation. *arXiv preprint arXiv:2608.02326*, 2026. URL https://arxiv.org/abs/2608.02326.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025. URL https://arxiv.org/abs/2510.04246.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Fei-Fei Li, Yuke Zhu, and Linxi Fan. RoboTTT: Context scaling for robot policies. *arXiv preprint arXiv:2607.15275*, 2026. URL https://arxiv.org/abs/2607.15275.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 267–274, 2002.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Shijie Lian, Bin Yu, Xiaopeng Lin, Zhaolong Shen, Laurence Tianruo Yang, Yurun Jin, Haishan Liu, Changti Wu, Hang Yuan, Cong Huang, and Kai Chen. IntentVLA: Short-horizon intent modeling for aliased robot manipulation. *arXiv preprint arXiv:2605.14712*, 2026. URL https://arxiv.org/abs/2605.14712.

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. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a. URL https://arxiv.org/abs/2604.04161.

Yushen Liang, Yue Peng, Baosheng Jin, Tianluo Zhang, Xinyu Zhang, Shuyi Zhou, Zhuoran Chen, Xinqi Liu, and Shenji Wan. TFP: Temporally conditioned memory-fusion policies for visuomotor learning. *arXiv preprint arXiv:2607.08283*, 2026b. URL https://arxiv.org/abs/2607.08283.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.03310.

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

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, Dequan Wang, and Yang Gao. Learning native continuation for action chunking flow policies. *arXiv preprint arXiv:2602.12978*, 2026. URL https://arxiv.org/abs/2602.12978.

Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6961–6971, 2020. URL https://proceedings.mlr.press/v119/misra20a.html.

Junnan Nie, Jiayi Li, Jiachen Zhang, Junyi Lao, Chenghao Liu, Tianle Zhang, Liang Lin, and Songfang Huang. PACE: Phase-aware chunk execution for robot policies with action chunking. *arXiv preprint arXiv:2606.00537*, 2026. URL https://arxiv.org/abs/2606.00537.

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. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2512.01809.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. In *Proceedings of the 38th Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pp. 5248–5351, 2025. URL https://proceedings.mlr.press/v291/simchowitz25a.html.

Satinder Singh, Michael R. James, and Matthew R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pp. 512–519, 2004.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.02502.

Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022. URL https://www.jmlr.org/papers/v23/20-1165.html.

Xiao-Ming Wu, Bin Fan, Kang Liao, Jian-Jian Jiang, Runze Yang, Yihang Luo, Zhonghua Wu, Wei-Shi Zheng, and Chen Change Loy. VLANeXt: Recipes for building strong VLA models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.18532.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Yihan Zeng, Minghao Ye, Yiyuan Chen, Yide Shentu, Philipp Wu, Zike Yan, and Zhongyu Li. KEMO: Event-driven keyframe memory for long-horizon robot manipulation with VLA policies. *arXiv preprint arXiv:2606.23589*, 2026. URL https://arxiv.org/abs/2606.23589.

Dijia Zhan, Xuemiao Xu, Jinyi Li, and Jie Tang. SEAM: Smooth execution of action-chunked motion for vision-language-action policies. *arXiv preprint arXiv:2607.04609*, 2026. URL https://arxiv.org/abs/2607.04609.

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. In *International Conference on Learning Representations*, 2026. 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. *arXiv preprint arXiv:2304.13705*, 2023. URL https://arxiv.org/abs/2304.13705.

A THEORY: WHEN A REQUERY INTERFACE IS SUFFICIENT

The theory identifies a relation that generation quality alone leaves unspecified: which physical branch receives which continuation. A 2×2 exchange isolates its value while preserving both marginals. The proof below characterizes the zero-interaction case, bounds departures from it, and carries the argument through later handoffs. The same rectangular contrast subsequently connects the fixed-bank diagnosis to the history-conditioned repair in Appendix F.

A.1 PAIRING IS HARMLESS EXACTLY UNDER ADDITIVE CONTINUATION VALUE

Fix the query input and continuation horizon and suppress their subscripts. Let $\mathcal{H} = \mathrm{supp}(\mu)$ and $\mathcal{Y} = \mathrm{supp}(\nu)$ be finite, with positive marginal mass and a well-defined continuation value $Q(h, y)$ on every cross-pair. If $Q(h, y) = a(h) + b(y)$, then for every $\gamma \in \Gamma(\mu, \nu)$,

$$\sum_{h,y} \gamma(h, y)Q(h, y) = \sum_{h} \mu(h)a(h) + \sum_{y} \nu(y)b(y), \tag{10}$$

which is independent of the coupling.

For the converse, assume the expectation is invariant. The product coupling $\gamma_0(h, y) = \mu(h)\nu(y)$ is strictly positive. Choose any $h, h' \in \mathcal{H}$ and $y, y' \in \mathcal{Y}$. For sufficiently small $|\epsilon|$, define γ_ϵ by adding ϵ to cells (h, y) and (h', y'), subtracting it from (h, y') and (h', y), and leaving all other cells unchanged. This preserves every row and column marginal, so $\gamma_\epsilon \in \Gamma(\mu, \nu)$. Invariance gives

$$0 = \frac{J(\gamma_\epsilon) - J(\gamma_0)}{\epsilon} = Q(h, y) + Q(h', y') - Q(h, y') - Q(h', y). \tag{11}$$

Thus every 2×2 interaction contrast vanishes. Fix (h_0, y_0) and define $a(h) = Q(h, y_0)$ and $b(y) = Q(h_0, y) - Q(h_0, y_0)$. Applying the vanishing contrast to $(h, h_0; y, y_0)$ yields $Q(h, y) = a(h) + b(y)$, proving the criterion. Subtracting the reference column y_0 also shows that this is equivalent to a history-independent relative continuation profile.

Because the finite coupling polytope $\Gamma(\mu, \nu)$ is compact and J is linear, its maximum and minimum are attained. If $Q(h, y) = a(h) + b(y)$, the calculation above makes J constant and the coupling diameter in equation 4 is zero. Conversely, zero diameter makes the maximum and minimum equal, hence J is constant over the entire polytope; the 2×2 argument above then recovers the additive form. This proves the zero-diameter criterion used in the main text.

The approximate guarantee in equation 5 follows from the same cancellation. For any $\delta > 0$, choose a, b whose residual $r(h, y) = Q(h, y) - a(h) - b(y)$ obeys $\|r\|_\infty \leq \varepsilon + \delta$. Two couplings with the same marginals have the same expectation of $a(H) + b(Y)$, and therefore

$$|J(\gamma) - J(\gamma')| = |\mathbb{E}_\gamma r - \mathbb{E}_{\gamma'} r| \leq 2(\varepsilon + \delta). \tag{12}$$

Taking the maximizing and minimizing couplings and then letting $\delta \to 0$ gives $\mathfrak{C} \leq 2\varepsilon$. The guarantee concerns a fixed handoff and horizon. The closure condition below supplies one route to sufficiency across later queries.

A.2 PER-QUERY MARGINALS CAN CONCEAL EXPONENTIAL TRAJECTORY GAPS

Let $M_1, \ldots, M_Q$ each have marginal π on $\{1, \ldots, K\}$. Under product coupling,

$$\Pr(M_1 = \cdots = M_Q) = \sum_{m=1}^{K} \Pr(M_1 = m, \ldots, M_Q = m) \tag{13}$$

$$= \sum_{m=1}^{K} \prod_{q=1}^{Q} \Pr(M_q = m) = \sum_{m=1}^{K} \pi_m^Q. \tag{14}$$

Under the diagonal coupling $\gamma(m_1, \ldots, m_Q) = \pi_m$ when $m_1 = \cdots = m_Q = m$ and zero otherwise, every marginal is still π and the constant-mode event has probability one. Thus identical query-wise marginals can yield different probabilities of maintaining a mode throughout the trajectory.

A.3 Model closure is sufficient through later queries

The main text defines continuation sufficiency directly through the horizon-indexed relative value needed for the fixed task and downstream policy. A standard stronger condition guarantees it for every finite-horizon risk-neutral policy that acts through an abstract query state. Let $S_q = \psi(H_q)$, assume the admissible candidate set is determined by S_q, and suppose that for every candidate y:

$$\mathbb{E}[R_q \mid H_q = h, Y_q = y] = \bar{r}(\psi(h), y), \tag{15}$$

$$\mathcal{L}(S_{q+1} \mid H_q = h, Y_q = y) = \bar{P}(\cdot \mid \psi(h), y). \tag{16}$$

With integrable additive rewards and a fixed terminal convention, Bellman induction shows that the finite-horizon value of every candidate is a function of (S_q, y). The base case follows from reward preservation. If the claim holds with $n-1$ queries remaining, the expectation of the next value depends only on the displayed pushforward kernel, establishing the n-query case.

Model closure is sufficient but stronger than the quotient required here. Its role is to expose why preserving an immediate ranking is not enough: a candidate also determines the next abstract state on which later queries act. Histories with different rewards or transitions can therefore remain value-equivalent under a particular downstream policy.

A.4 Earlier queries choose where later differences are evaluated

The performance bridge in equation 7 makes the recursive dependence explicit. Consider two possibly time-inhomogeneous requery mechanisms A and B on complete handoff histories, with a common initial distribution, integrable additive rewards, and the same terminal convention. Define $V^B_{Q+1} = 0$ and $V^B_q = \mathcal{B}^B_q V^B_{q+1}$. Along a trajectory generated by A,

$$J^A - J^B = \sum_{q=1}^{Q} \mathbb{E}_A \left[R_q + V^B_{q+1}(H_{q+1}) - V^B_q(H_q) \right] \tag{17}$$

$$= \sum_{q=1}^{Q} \mathbb{E}_{H_q \sim d^A_q} \left[\left((\mathcal{B}^A_q - \mathcal{B}^B_q) V^B_{q+1} \right)(H_q) \right]. \tag{18}$$

The first line telescopes because the initial distribution is shared and the terminal value is fixed; conditioning on H_q gives the second line. Here $\mathbb{E}_A$ averages trajectories generated by mechanism A.

Now condition a query-q term on $I_q = i$ under d^A_q, and set $\mu_i = d^A_q(\cdot \mid I_q = i)$. Apply each mechanism's candidate kernel to this same history distribution. If the resulting candidate marginals agree at ν_i, the two joint laws belong to $\Gamma(\mu_i, \nu_i)$. This comparison uses B's kernel at histories reached by A, not B's own rollout distribution; d^A_q and d^B_q need not coincide. With Q^B_{Q-q+1} including the segment reward and subsequent value under B, the fiber-averaged term is $J_{i,Q-q+1}(\gamma^A_i) - J_{i,Q-q+1}(\gamma^B_i)$ and is bounded in magnitude by $\mathfrak{C}_{i,Q-q+1}$. Averaging over occupied fibers, summing over queries, and applying the triangle inequality gives the main-text recursive bound. The approximation in equation 5 replaces each diameter by 2ε. Thus the standard identity becomes a requery-specific bridge: the local pairing sensitivity at query q is weighted by the handoff distribution produced by the preceding queries.

The criterion becomes testable by executing one candidate bank on different physical receivers. Holding the bank fixed exposes whether changing the receiver changes candidate-relative value. The following diagnosis applies this test first to controlled branches and then to observation-matched states reached during natural execution.

B Branch-Conditioned Value in Controlled and Natural Execution

The diagnosis tests the relation that execution context needs to resolve: which candidate fits the receiving physical branch. Controlled branch crossing identifies this relation and its variation across tasks and stages. Observation-matched natural trajectories then show it in states reached without a constructed counterbranch.

B.1 CANDIDATE VALUE DEPENDS ON THE RECEIVING BRANCH

A candidate's local value is not intrinsic to the candidate: the same frozen bank changes ordering and value when executed on a different controlled physical branch. We compare observation-only Transformer–Flow with eight integration steps and U-Net–Diffusion with 16 DDIM steps. Each uses 50 tasks, eight on-policy and eight expert-coverage handoffs per task, and 16 fixed-noise candidates per handoff. Both models generate 32-action chunks from two-frame observation windows; the diagnostic evaluates a 16-action execution from each candidate. Each bank is generated once and cross-executed from the reference branch, a same-input controlled counterbranch, and an exact-replay snapshot. The study contains 800 groups per generator and 79,200 short executions including deterministic controls. Uncertainty is estimated by resampling tasks.

Constructing the receiving branches. At each anchor, we save MuJoCo's integration state, the environment's cached observations and episode counters, and its random-generator state. The reference and replay arms restore that snapshot. For the counterbranch, we restore it, replace the full generalized velocity vector by its negative, $\dot{q} \leftarrow -\dot{q}$, and call `mj_forward` to recompute derived dynamics before saving the new snapshot. Positions are unchanged; the operation does not separately edit actuator controls, activation state, or mocap targets. The snapshot checks verify preserved positions and current proprioception, reversed nonzero finite velocities, and reference/replay execution agreement. All 1,600 selected anchors are retained: 400 each from Flow, Diffusion, expert and ACT trajectories, with the expert anchors shared by both generators. Five exceed the source-derived velocity envelope (four Flow, one ACT); envelope exceedance is not an exclusion in the reported evaluation.

Anchors are selected before candidate evaluation from 16 source episodes per task and coverage. On-policy selection prioritizes distinct episodes; expert selection takes up to two anchors from each of the four phases, then fills missing slots from the remaining pool. A fixed outcome-independent ordering breaks ties. Each anchor follows at least one executed prefix and leaves room for the 16-action evaluation.

The generator receives the reference trajectory's cached two-frame RGB and proprioceptive window, with its task conditioning and normalization. The bank is generated once from this input and executed on both receivers. The counterbranch therefore changes physical motion while retaining the reference observation history: it is a controlled test of continuation sensitivity, rather than a second naturally realized two-frame history.

For branch b and candidate j, let R_{bj} be cumulative return over the next 16 executed actions and $r_b(j)$ its within-bank rank. The two Figure 2 diagnostics are

$$D_{\mathrm{rank}} = \frac{\sum_{j=1}^{K} |r_+(j) - r_-(j)|}{K(K-1)}, \qquad g(R) = \frac{\frac{1}{2}\sum_b \max_j R_{bj} - \max_j \frac{1}{2}\sum_b R_{bj}}{\mathrm{IQR}(R) + 10^{-8}}.$$

We report $G_{\mathrm{excess}} = g(R_{\mathrm{physical}}) - g(R_{\mathrm{replay}})$. The first quantity is mean absolute rank displacement divided by $K-1$, displayed as a percentage. It scales each candidate's movement by the full rank span, not by the maximum distance between permutations (at $K = 16$, complete rank reversal gives 53.33%); the second measures the normalized value of a branch-aware finite-bank oracle. The coupling diameter below instead measures value variation at fixed marginals: its optimizing assignment must preserve candidate frequencies, whereas an oracle may select a different best candidate per branch. The rank statistic assigns ordinal positions by double argsort; different-best uses argmax. Exact ties therefore inherit the implementation's index convention, rather than averaged ranks. The value-based gap and diameter are reported alongside these ordering summaries.

On policy trajectories, the unexecuted suffix is also available as a candidate. The augmented-bank gap applies g to this suffix together with the generated bank, using their combined return IQR for normalization.

The crossed matrices also instantiate the coupling diameter directly. With uniform marginals over two branches and $K = 16$ candidates, let $d_j = Q(h^+, y_j) - Q(h^-, y_j)$. The exact transport solution assigns the largest half of the d_j values to h^+ and the smallest half to h^-, giving

$$\widehat{\mathfrak{C}} = \frac{1}{K}\left(\sum_{j \in \mathrm{top}_{K/2}} d_j - \sum_{j \in \mathrm{bottom}_{K/2}} d_j\right).$$

Table 3: **Branch-conditioned value across coverage and model structure.** Top: the canonical Transformer–Flow and U-Net–Diffusion policies on on-policy and expert handoffs. Bottom: all four models on the same 400 expert handoffs. Different-best is the fraction of handoffs whose maximizing candidate changes across branches. The bank-only gap is G_{excess}; the augmented gap includes the old suffix. Intervals are task-bootstrap 95% CIs.

Generator	Coverage	Bank-only gap	Augmented gap	Different-best
Flow	On-policy	.00864 [.00444, .01358]	.00831 [.00423, .01314]	17.25% [12.75, 22.25]
Flow	Expert	.01325 [.00498, .02417]	—	—
Diffusion	On-policy	.00642 [.00256, .01129]	.00643 [.00271, .01113]	13.50% [9.50, 17.76]
Diffusion	Expert	.01224 [.00570, .02001]	—	—

Backbone	Generator	Different-best on shared expert support [95% CI]
Transformer	Flow	19.50% [14.74, 24.75]
Transformer	Diffusion	20.00% [15.75, 24.25]
U-Net	Flow	21.00% [15.75, 26.50]
U-Net	Diffusion	18.00% [13.25, 23.00]

After normalizing each handoff by its observed $2 \times K$ value range, the task-balanced means are $.02689/.03424$ for Flow on-policy/expert coverage and $.02156/.03322$ for Diffusion. The diameter is positive in respectively $48/48/48/49$ of 50 tasks; all replay diameters are zero.

Candidate ranking is broadly branch dependent: every MT50 task shows nonzero rank divergence in at least one generator–coverage cell, and $44/50$ do so in all four. The value differences vary in magnitude: $46/50$ tasks show a positive replay-subtracted value gap in at least one cell, but only $14/50$ do so in all four; the five largest contributors account for 62–75% of each aggregate. Thus branch crossing exposes a broad relational change in which continuation is preferred, with its control cost amplified in specific tasks and stages.

B.2 Two candidates already expose branch-dependent choice

The witness already appears when each decision offers only two alternatives and persists as the choice set grows. The same frozen 16-candidate matrices define the estimands over every $K = 2$ and $K = 4$ subset, 512 outcome-independent $K = 8$ subsets, and the full $K = 16$ bank. The same subset schedule is used at every anchor, with all comparisons computed from the original value matrices.

Table 4: **Branch-dependent choice persists across bank sizes.** Task-balanced on-policy results. Selection gain is the unnormalized return advantage of branch-aware choice from the old suffix plus the frozen bank over one branch-independent choice.

Generator	Metric	$K = 2$	$K = 4$	$K = 8$	$K = 16$
Flow	Different-best (%)	8.51	12.97	15.93	17.25
Flow	Selection gain	.00723	.00784	.01057	.01370
Diffusion	Different-best (%)	6.31	10.22	12.38	13.50
Diffusion	Selection gain	.01303	.01442	.01425	.01506

Larger banks create more opportunities for the best candidate to differ between branches, but two alternatives already expose the interaction. Selection gain measures the value available to a branch-aware oracle over this finite set.

Neither always regenerating nor always retaining the old suffix supplies that branch-aware choice. Task-mean fresh-minus-old return is positive on 25 Flow, 24 Diffusion and 26 deterministic action-chunking tasks, versus 23, 24 and 22 negative tasks, respectively, with two ties in each comparison. The advantage lies in selecting according to the receiver, not in assigning a fixed preference to a fresh

or previously generated candidate. The deterministic comparator is LeRobot ACT with a ResNet-18 visual encoder and a 512-wide, eight-head Transformer (four encoder layers, one decoder layer). It uses one RGB/proprioceptive observation, generates 32 actions and commits 16. Training combines action ℓ_1 reconstruction with a 32-dimensional variational latent and KL weight 10. Inference fixes the latent to zero and uses no temporal ensembling for this fresh-versus-suffix comparison.

B.3 Task geometry and stage expose different sources of heterogeneity

Branch-dependent ordering is broad, but its immediate control cost is structured by task geometry and execution stage. Terminal-recovery groups distinguish tasks by their interaction constraints. Within-task comparisons locate changes in branch sensitivity during interaction (Figure 4).

The task partition describes terminal interaction geometry, not overall task difficulty. Low-recovery-margin tasks require a localized insertion, press, lock, constrained placement, or tool contact. High-recovery-margin tasks permit a broader reach, push, sweep, or large-stroke articulated motion. Labels were assigned from task semantics without using crossed-execution outcomes. Table 5 lists the complete partition and the subset eligible for the within-task stage contrast.

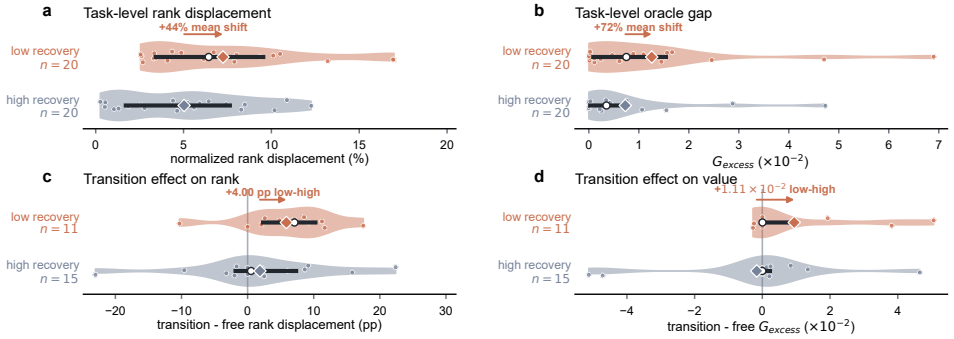


Figure 4: **Task and stage structure of branch-relative continuation value. (a–b)** Task means across the four canonical generator–coverage cells for 20 low- and 20 high-recovery-margin tasks; 10 mixed tasks remain in the full analysis. **(c–d)** Within-task transition-minus-free contrasts for the 11 low-margin and 15 high-margin tasks with both stage types, averaging the two canonical generators on shared expert support. Small points are tasks, diamonds means, open circles medians, and thick segments interquartile ranges.

The 10 mixed cases excluded from the low–high recovery contrast are listed below. They remain in the all-MT50 identification results.

```
basketball-v3              bin-picking-v3
plate-slide-back-side-v3   plate-slide-back-v3
plate-slide-side-v3        plate-slide-v3
soccer-v3                  stick-pull-v3
stick-push-v3              sweep-into-v3
```

Cross-cell mean rank divergence is .0725 versus .0503 for low- and high-recovery tasks, respectively (44% higher). This direction survives leave-one-task-out checks, removing the top five tasks, and all 1,024 assignments of the mixed group.

The within-task analysis asks when that sensitivity changes along execution. Stage labels use the preceding 16 frames of contact, object-state motion and gripper motion. The label priority is release/contact transition, onset, sustained interaction, then free motion. Release detects a loss of contact between the two half-windows or a decrease in the gripper coordinate; onset detects new contact, object motion crossing its threshold, or gripper motion above its threshold. Sustained interaction requires contact and object motion; free motion requires neither contact nor above-threshold motion. Other frames remain unclassified. Two separate discovery episodes per source and task set the positive-motion thresholds at the 60th percentile for object motion and 75th for gripper change (3.27×10^{-4} and 4.19×10^{-4} in raw state coordinates). These labels use past execution, not the return of the candidates being evaluated.

Table 5: **Complete MT50 terminal-recovery partition and stage support.** "Paired stage" indicates that the eight shared expert handoffs contain at least one free-motion anchor and at least one pooled onset or release/contact transition anchor, so the within-task difference in Figure 4c–d is defined. A "no" entry denotes missing paired-stage support.

Low terminal recovery margin ($n = 20$)	Paired stage?	High terminal recovery margin ($n = 20$)	Paired stage?
assembly-v3	no	coffee-pull-v3	yes
box-close-v3	no	coffee-push-v3	yes
button-press-topdown-v3	yes	disassemble-v3	yes
button-press-topdown-wall-v3	yes	door-close-v3	yes
button-press-v3	no	door-open-v3	no
button-press-wall-v3	yes	drawer-close-v3	yes
coffee-button-v3	yes	drawer-open-v3	yes
dial-turn-v3	yes	faucet-close-v3	yes
door-lock-v3	yes	faucet-open-v3	yes
door-unlock-v3	yes	handle-pull-side-v3	no
hammer-v3	no	handle-pull-v3	no
hand-insert-v3	no	peg-unplug-side-v3	yes
handle-press-side-v3	yes	push-back-v3	yes
handle-press-v3	yes	push-v3	no
lever-pull-v3	no	push-wall-v3	no
peg-insert-side-v3	yes	reach-v3	yes
pick-out-of-hole-v3	no	reach-wall-v3	yes
pick-place-v3	no	sweep-v3	yes
pick-place-wall-v3	no	window-open-v3	yes
shelf-place-v3	yes	window-close-v3	yes

Comparing transition and free-motion handoffs within each task holds task mix fixed. The shared expert cohort supplies both stage types for 11 low- and 15 high-recovery tasks out of 20 in each group. Within-task stage contrasts have mixed signs. The low-minus-high phase interaction has the same observed direction in all four Flow/Diffusion by on-policy/expert cells for both rank divergence and the finite-bank value gap. The paired-stage cohort in the main figure makes that mean pattern visible while retaining the large task spread.

Table 6: **Stage effects depend on task recovery margin.** Each interaction is $(\mathrm{transition} - \mathrm{free})_{\mathrm{low}} - (\mathrm{transition} - \mathrm{free})_{\mathrm{high}}$. The upper panel uses each generator–coverage cell's paired-stage tasks. The lower panel is the canonical two-generator average on the 11 low- and 15 high-recovery tasks sharing expert stage support in the main figure; intervals resample paired tasks.

Interaction	Flow on-policy	Flow expert	Diffusion on-policy	Diffusion expert
Excess value gap	$+.00516$	$+.01835$	$+.01960$	$+.00382$
Rank divergence	$+.00325$	$+.03125$	$+.06851$	$+.04879$

Metric	High-recovery change	Low-recovery change	Low − high [95% CI]
Rank displacement (pp)	$+1.86$	$+5.86$	$+4.00$ $[-2.73, 10.48]$
Oracle gap ($\times 10^{-2}$)	$-.165$	$+.943$	$+1.108$ $[-.327, 2.724]$

Terminal geometry and execution stage describe different aspects of the same decision. A constrained task can allow broad freedom during approach, then make the continuation more consequential at contact. The larger mean stage increase in the low-recovery group follows this pattern, with mixed task-level signs and intervals spanning zero. The paired distributions show where this pattern appears and how much it varies, rather than treating all handoffs in a task as equally sensitive.

B.4 Natural execution also produces branch-dependent preferences

Branch-dependent candidate value also appears on states reached by ordinary policy execution. The controlled construction above isolates the relation by holding the visible input fixed; the natural comparison asks whether similar visible contexts reached along different trajectories also admit different preferred continuations. Table 7 shows this effect for both Transformer–Flow and U-Net–Diffusion on naturally reached branches.

Natural branches and observation matching. For each model family, we collect eight paired rollout slots on each of the 50 Meta-World tasks using the observation-only and execution-conditioned policies, giving 800 trajectories per family. Each slot shares the task and environment seed between its two source policies. Both use the canonical seed-42, 60k checkpoints and two-frame corner2 RGB observations at 128×128. At each query, each source policy generates 32 actions and executes 16 before querying again with fresh observations. Flow uses eight Euler steps and Diffusion uses DDIM16. The execution-conditioned source accumulates realized transitions from the start of its trajectory. Both policies supply physical receivers; each fixed bank is evaluated on both states.

We pair boundaries at the same noninitial query index and the same execution phase, with at least 16 steps remaining in each trajectory's 500-step budget. Phase labels follow the free-motion, interaction-onset, sustained-interaction and release/contact-transition definitions above; unclassified boundaries are retained as a separate category. A pair is accepted when the RGB root-mean-square difference over the two-frame window is at most $.0196354$ and the maximum absolute difference in normalized proprioception is at most $.0897458$. Each threshold is the lower quartile of its observation-distance statistic over four pilot calibration pairs, capped in advance at $.03$ for RGB and $.5$ for proprioception. These thresholds are then held fixed for both formal model cohorts, without selecting on counterfactual return or success. The resulting rates describe the accepted, approximately observation-matched states, not the unconditional incidence of mismatch across deployment.

Fixed candidates and branch-relative value. Each accepted pair supplies two 16-candidate banks, generated by the observation-only model once from each receiver's actual observation window. The two banks use common noise indices. Each bank is then held fixed and executed on both receiving states and a repeat of the first state, yielding 48 short executions per bank, with 16 actions per execution. Both actual observations contribute candidates, while every value comparison evaluates the very same actions on the two receivers.

We sum rewards over the 16-action execution and reuse the rank-divergence and normalized, replay-subtracted finite-bank gap defined above. A strict winner change requires disjoint maximizing candidate sets on the two receivers, using an absolute return tolerance of 10^{-8}, so breaking a tie differently does not count as a change. Reference repeats yield identical rewards and final states in every bank, separating branch differences from replay variation.

Table 7: **Natural branches change the preferred continuation.** Task/slot-weighted means with 95% hierarchical bootstrap intervals. Rank displacement and winner change are percentages; the value gap uses the controlled diagnostic's normalization.

	Transformer–Flow	U-Net–Diffusion
Natural trajectories	800	800
Joint eligible query boundaries	3,674	4,435
Same-phase boundaries	2,247	2,625
Observation-matched pairs	589	291
Supported tasks / attempted tasks	49/50	41/50
Fixed candidate banks	1,178	582
Short executions	56,544	27,936
Tasks with a strict winner change	41/49	27/41
Rank displacement (%)	9.24 $[7.12, 11.66]$	10.46 $[7.37, 14.00]$
Strict winner change (%)	28.80 $[20.92, 37.17]$	22.55 $[13.69, 32.71]$
Excess value gap	$.01895$ $[.00904, .03436]$	$.01720$ $[.00543, .03345]$

Aggregation and matching sensitivity. We average the two banks and eligible queries within each rollout slot, then average slots within a task and supported tasks equally. Unsupported tasks remain in the coverage counts rather than receiving a zero effect. Intervals use 20,000 hierarchical bootstrap draws over tasks and slots, retaining the within-slot grouping of banks. Consequently, the headline winner-change rates differ from the raw bank counts of 317/1,178 and 157/582.

Tighter matching preserves branch dependence (Table 8). Halving both thresholds leaves 42 supported Flow tasks and 24 Diffusion tasks, with strict winner-change rates of 28.65% and 13.50%, respectively. The positive value gaps show that the result is more than a reshuffling of nearly equivalent ranks:

selecting according to the receiver offers higher mean short-horizon value within the same candidate bank. The stricter subsets have different task and slot coverage, so their magnitudes describe those subsets rather than a paired effect of changing the threshold.

Table 8: **Branch dependence persists as observation matching tightens.** Both RGB and proprioceptive thresholds are multiplied by the listed factor. All effects use the same task/slot weighting. Brackets give 95% intervals for the value advantage of receiver-specific selection.

Model	Factor	Tasks	Banks	Winner change (%)	Excess value gap
Transformer–Flow	.50	42	572	28.65	.02844 [.00750, .06737]
	.75	49	912	31.55	.02552 [.01085, .04941]
	1.00	49	1,178	28.80	.01895 [.00904, .03436]
U-Net–Diffusion	.50	24	208	13.50	.00668 [.00031, .02052]
	.75	37	346	19.78	.01172 [.00293, .02467]
	1.00	41	582	22.55	.01720 [.00543, .03345]

Task and stage resolution. Assembly provides a concrete natural counterpart to the controlled diagnosis. Its preferred candidate changes on 8 of 12 Flow banks across five rollout slots and on 12 of 24 Diffusion banks across six slots; the slot-weighted rates are 65.0% and 55.6%. All accepted Flow assembly pairs and 11 of the 12 Diffusion pairs have the release/contact-transition phase label. These naturally reached states expose branch-dependent choice within a particular stage of assembly.

Across all tasks, accepted banks cover free motion, onset, sustained interaction, release/contact transition and unclassified boundaries in counts $224/154/40/656/104$ for Flow and $44/64/14/382/78$ for Diffusion. This coverage is uneven and is dominated by release/contact transitions; pooled phase counts combine different tasks. The controlled within-task comparisons above identify the stage pattern, while the natural audit establishes its broader setting: trajectories reached without a constructed counterbranch can still require different choices among the same candidate continuations.

C Control Gains Across Architectures and Training Seeds

Execution conditioning improves complete-policy performance across backbone–generator pairings. Independent training runs and increased width test the stability of this gain. Comparing immediate candidate value with full-rollout success then clarifies what changes in the trained policies before we examine how they use execution records.

C.1 The complete cross improves across independent training seeds

Stateful Requery improves three-seed mean success in every benchmark–cell combination. Table 9 separates the individual runs behind Table 1. Each score averages tasks equally, with 50 rollout slots per task. Seeds 42, 43, and 44 denote independent training runs; the mechanism comparisons use the seed-42 checkpoints rather than a mixture of these policies.

The gain persists across independent training runs. On LIBERO, the Transformer–Diffusion observation-only policy ranges from 52.76 to 53.69%, and the execution-conditioned policy improves at every seed. The task-level pattern is also reproducible: in Transformer–Flow, U-Net–Flow, and U-Net–Diffusion, respectively, 83, 74, and 47 of 90 LIBERO tasks improve in all three seeds. Thus the execution-conditioned interface improves both aggregate success and a broad set of individual tasks across independent training runs.

C.2 Execution conditioning complements action-side capacity

Under the shared LIBERO endpoint, ordinary Flow rises from 17.22 to 26.67% when hidden width, attention heads, and FFN width double. Applying the same complete-width scaling to Stateful Flow changes 67.96 to 68.62%. At equal width, the execution-conditioned policy remains 41.96 points ahead (computed before rounding the displayed success rates). Capacity helps the ordinary generator, while the execution-conditioned advantage survives this increase.

Table 9: **The complete structural cross across three training seeds.** Task-balanced success (%) with 50 rollout slots per task on Meta-World MT50 and LIBERO-90.

		Flow			Diffusion		
Backbone	Seed	Observation	Stateful	Δ	Observation	Stateful	Δ
			Meta-World MT50				
	42	71.96	76.44	+4.48	51.04	53.16	+2.12
Transformer	43	71.92	77.84	+5.92	49.44	49.64	+0.20
	44	75.88	79.60	+3.72	51.00	52.36	+1.36
	42	78.04	79.32	+1.28	64.28	70.44	+6.16
U-Net	43	80.32	80.32	+0.00	66.12	72.04	+5.92
	44	78.84	80.32	+1.48	65.24	70.20	+4.96
			LIBERO-90				
	42	17.22	67.96	+50.73	53.69	66.22	+12.53
Transformer	43	33.53	63.60	+30.07	53.18	68.91	+15.73
	44	22.27	71.38	+49.11	52.76	74.71	+21.96
	42	31.49	78.98	+47.49	69.76	84.40	+14.64
U-Net	43	42.73	83.11	+40.38	67.71	82.09	+14.38
	44	36.58	80.58	+44.00	66.96	83.31	+16.36

C.3 Immediate candidate value differs from complete-policy performance

The trained policies improve complete trajectories without consistently producing higher-return chunks at shared expert handoffs. Each policy generates its own candidate bank at 400 such handoffs per cell (Table 10). Mean 16-action return is lower for Stateful in three cells and higher in the fourth, while full-rollout success improves in all four. The short-segment comparison measures immediate progress from shared handoffs; complete trajectories also include the states and corrective actions produced by subsequent queries. Higher rollout success can therefore coexist with lower first-segment return. The contrast directs attention to how execution records shape successive decisions, beyond the quality of one candidate bank.

Table 10: **Local candidate quality and complete-rollout improvement.** Both columns are Stateful minus observation-only at seed 42. Local return averages each model's bank at the same 400 expert handoffs; success evaluates each policy's complete trajectories.

Backbone	Generator	Local return change	Rollout gain (pp)
Transformer	Flow	-1.3119	$+4.48$
Transformer	Diffusion	-0.2065	$+2.12$
U-Net	Flow	-0.6947	$+1.28$
U-Net	Diffusion	$+0.1312$	$+6.16$

D Committed Execution Is the Carrier

The useful content of execution links what the robot attempted to what became physical. Command records and realized-transition records test which part of that link benefits control. A separate lifecycle comparison asks whether the latest realized motion suffices, or earlier commitments also help.

D.1 Carrier utility depends on the learned interface

Realized transitions provide a consistently strong carrier across the four model families. Table 11 expands the within-family gains in Table 2 into absolute trained endpoints. The carriers supply the latest executed commands, accumulated commands, or realized transitions.

Both realized-transition designs improve over observation-only conditioning in all four families, whereas both command carriers reduce success. The no-time realized-transition carrier ranks in the top two in every family. Its advantage over the full record in three families also shows that a

Table 11: **Control performance across execution carriers.** Task-balanced Meta-World MT50 success (%) at seed 42. Each carrier is compared with the observation-only policy in its column.

	Transformer		U-Net	
Interface content	Flow	Diffusion	Flow	Diffusion
Observation only	71.96	51.04	78.04	64.28
Latest executed commands	53.72	36.44	57.08	49.36
Accumulated executed commands	56.88	43.72	57.80	48.88
Realized transitions; no elapsed time	79.60	51.88	79.96	72.08
Stateful Requery (full record)	76.44	53.16	79.32	70.44

simpler encoding can retain useful execution information. The consistent distinction is what the record describes: commands specify intended motion, while realized transitions capture the response that later actions must continue.

The gain does not require a matrix carrier. A matched Transformer–Flow control replaces the KDA carrier with an episode-local GRU that reads the same chronological stream of committed transitions. It reaches 78.04% success ($1{,}951/2{,}500$), compared with 76.44% for the KDA carrier and 71.96% for observation-only conditioning. The GRU is updated only by executed events and resets at the start of each episode. Generic recurrence therefore recovers the control gain; the evidence supports persistent, execution-grounded conditioning rather than a mechanism unique to KDA.

The aggregate differences arise from uneven task responses. For Transformer–Diffusion, accumulated commands lower mean success by 7.32 points, with 14 tasks improving, 11 tied, and 25 declining. U-Net–Flow's no-time realized record gains 1.92 points, with 19 improving, 17 tied, and 14 declining. Recording the robot's response improves the aggregate comparison without imposing a common response on every task.

D.2 Earlier commitments matter beyond the latest realized prefix

In Flow, the most recent realized prefix does not exhaust the value of committed execution. Both lifecycle arms use the same two-frame observation interface and an isomorphic writer/read path. Latest-prefix clears its carrier before writing its newest executed prefix; Persistent writes its own newest prefix without erasing earlier commitments. Each arm is trained under its deployment lifecycle.

Table 12: **Earlier commitments retain value beyond the latest prefix in Flow.** Task-balanced Meta-World MT50 success (%) for separately trained lifecycle designs. Intervals resample tasks.

Policy	Latest prefix	Persistent	Δ pp [95% CI]
Transformer–Flow	58.52	76.44	$+17.92$ $[+10.64, +25.76]$
U-Net–Diffusion	69.80	70.44	$+0.64$ $[-1.68, +3.36]$

The Flow comparison establishes that earlier commitments contribute beyond the latest motion: both designs encode their latest realized prefix, yet preserving what preceded it improves full-rollout success. Positive task signs occur on $25/50$ tasks (9 negative and 16 tied). Diffusion has positive signs on $13/50$ tasks (18 negative and 19 tied), with a small aggregate difference whose interval includes zero.

D.3 Prediction aggregation and execution conditioning are distinct operations

Temporal ensembling retains earlier predictions; an execution record retains what happened after actions were applied. The former combines proposed continuations, while the latter changes the information used to generate them. This distinction explains how aggregation can help without resolving every branch-dependent choice, and why it can be combined with Stateful Requery.

What averaging improves. Let $u_t^{(j)} \in \mathbb{R}^d$ be predictions made at different queries for the same physical execution time t. Temporal ensembling forms $\bar{u}_t = \sum_j w_j u_t^{(j)}$, with $w_j \geq 0$ and $\sum_j w_j = 1$. For any common target action $u_t^\star$, the exact identity

$$\sum_j w_j \|u_t^{(j)} - u_t^\star\|^2 - \|\bar{u}_t - u_t^\star\|^2 = \sum_j w_j \|u_t^{(j)} - \bar{u}_t\|^2 \geq 0 \tag{19}$$

follows by expanding the squares and using $\sum_j w_j (u_t^{(j)} - \bar{u}_t) = 0$. Averaging therefore removes the disagreement component of squared error relative to the weighted average of individual errors. This is an action-estimation property, not an ordering of physical continuation values: those values need not be concave in action.

What aggregation cannot recover. Let Z_t contain all inputs available to an aggregation rule, including the overlapping predictions and their weights. Suppose two physical branches b_+ and b_- induce the same conditional law of Z_t. Any common aggregation rule, represented by a kernel $K(du \mid z)$, then satisfies

$$P(du \mid b_+) = \int K(du \mid z) P(dz \mid b_+) = \int K(du \mid z) P(dz \mid b_-) = P(du \mid b_-). \tag{20}$$

Thus aggregation preserves this unresolved branch distinction. For a concrete value consequence, consider equally likely branches whose correct actions are v and $-v$, with loss $\|u - v\|^2$ and $\|u + v\|^2$, respectively. Any common output law with finite second moment incurs mean loss

$$\tfrac{1}{2}\mathbb{E}\|U - v\|^2 + \tfrac{1}{2}\mathbb{E}\|U + v\|^2 = \mathbb{E}\|U\|^2 + \|v\|^2 \geq \|v\|^2. \tag{21}$$

A record identifying the branch permits the appropriate action and zero loss. For $Q(b, u)$ equal to negative loss, the rectangular interaction of Section 2 on actions $v, -v$ is $8\|v\|^2$. The missing distinction therefore has precisely the branch–action value structure studied in this paper.

The premise concerns the full aggregation input, not just the current image: past predictions can themselves preserve branch information. Execution conditioning supplies a different information path, through realized transitions, and may separate branches that prediction aggregation leaves indistinguishable. Neither operation prevents use of the other.

A fixed-query-rate comparison. We apply overlap averaging to observation-only and Stateful policies at $H = 32$, $E = 16$, keeping their generation schedules unchanged. After the first query, the old unexecuted suffix and fresh prefix predict the same next 16 execution times. For each such time we execute

$$\bar{u}_t = \frac{u_t^{\text{old}} + e^{-0.01} u_t^{\text{fresh}}}{1 + e^{-0.01}}. \tag{22}$$

The first prefix is unaveraged, and each query retains its raw suffix for the next overlap. Stateful writes the transitions actually produced by the averaged commands. This tests prediction aggregation at the paper's standard execution interval, rather than ACT's every-step query schedule (Zhao et al., 2023). Each entry uses the seed-42 policy and 50 episodes on each of the 50 Meta-World tasks.

Table 13: **Execution conditioning remains useful with prediction aggregation.** MT50 success (%). The final column compares the two policies with temporal ensembling (TE) applied to both.

	Without TE		With TE		
Model	Observation	Stateful	Observation	Stateful	Gain (pp)
Transformer–Flow	71.96	76.44	75.68	78.04	+2.36
U-Net–Flow	78.04	79.32	78.32	79.40	+1.08
U-Net–Diffusion	64.28	70.44	62.60	68.32	+5.72

Overlap averaging improves both Transformer–Flow policies, changes U-Net–Flow little, and lowers both U-Net–Diffusion scores. Stateful retains a positive advantage under aggregation in each tested combination. These results distinguish conditioning generation on execution from combining its outputs; they do not require averaging itself to improve every policy.

E RETAINING AND UPDATING EXECUTION CONTEXT

An execution record must remain useful as new observations and new actions arrive. Holding the trained policy fixed, retention controls test how long its record contributes to control. Updated and stale records then distinguish preserving earlier execution from incorporating the robot's latest response.

E.1 RETENTION VALUE DEPENDS ON HOW THE DEPLOYED POLICY USES HISTORY

Retaining history improves all four policies relative to erasing their record (Table 14). Native execution retains accumulated state throughout the episode; reset presents zero state at every requery. The first query is shared, before either policy has committed any action. Thus the benefit of training an execution-conditioned interface and the value of continually reading its accumulated state are related but distinct questions.

Table 14: **Native and reset execution across the four model families.** Task-balanced MT50 success (%); each row compares one fixed Stateful checkpoint over 50 tasks and 50 paired slots per task. Intervals resample tasks. These deployment interventions do not retrain the policies.

Backbone	Generator	Native	Reset each query	Native − reset, pp [95% CI]
Transformer	Flow	76.44	49.76	$+26.68$ $[17.60, 35.96]$
Transformer	Diffusion	53.16	35.76	$+17.40$ $[9.00, 26.36]$
U-Net	Flow	79.32	23.88	$+55.44$ $[45.76, 64.72]$
U-Net	Diffusion	70.44	26.04	$+44.40$ $[33.08, 55.32]$

Retention duration and commitment length probe different parts of requery. At fixed $E = 16$, clearing state every one, two, or four queries, or retaining it for the episode, varies history depth. All four policies improve across these settings. The upper panel of Table 15 instead varies the execute horizon on the same 24 tasks. At $E = 1$, a fresh sample follows every action, leaving no multi-action chunk seam to repair.

Table 15: **Short commitments and retention duration probe complementary parts of requery.** Entries are native-minus-reset gains (pp) in the upper panel and gains over one-query retention in the lower panel. The execute-horizon comparison uses 24 tasks with 16 slots each; the retention comparison uses 50 tasks with 50 slots each.

	Execute horizon: actions before the next query			
Policy	$E = 1$	$E = 4$	$E = 8$	$E = 16$
Transformer–Flow	$+42.19$	$+42.97$	$+39.32$	$+25.52$
U-Net–Diffusion	$+66.41$	$+70.31$	$+64.58$	$+53.39$
	Retention duration at fixed $E = 16$			
Policy	1 query	2 queries	4 queries	Episode
Transformer–Flow	0.00	$+8.16$	$+17.84$	$+27.08$
Transformer–Diffusion	0.00	$+9.92$	$+13.28$	$+17.28$
U-Net–Flow	0.00	$+24.72$	$+46.64$	$+55.36$
U-Net–Diffusion	0.00	$+19.08$	$+36.88$	$+44.68$

These one-action native-over-reset gains show that the record matters even without a multi-action seam. Separately trained observation-only policies score 72.92% and 81.51% at $E = 1$, compared with 71.35% and 78.13% for Stateful Flow and Diffusion, respectively. Thus the trained models' dependence on their record differs from their advantage over another policy. The retention curves show that earlier execution remains useful across several feedback updates. Together, these results distinguish the record's value at the next action from the duration of its useful influence.

E.2 Execution records change generation and local continuation

Erasing execution state changes generation in every structural cell. The four-cell comparison uses 400 shared expert handoffs per model and holds the physical input, weights, and candidate noise fixed. Table 16 measures the resulting first-action displacement. This broad response establishes that the execution channel is active; physical continuation tests determine how it is used.

Table 16: **All four learned generators respond to execution state.** Native-versus-zero reader differences in the first executed action, in dimensionless environment command coordinates after denormalization and clipping to $[-1, 1]$. At each handoff, RMS averages over 16 paired draws and all four command channels (three translations and gripper); the table averages over 400 expert handoffs per cell.

Backbone	Flow	Diffusion
Transformer	.3595	.3754
U-Net	.5670	.5637

The Flow reader also distinguishes the chronology and source of that record. On 3,072 held-out queries, reconstructing the recorded history gives loss $.01859$, compared with $.43619$ after erasure, $.29955$ after reversal, and $.50404$ after cross-task substitution. The generator distinguishes the recorded sequence from reversed and cross-task histories: both the order and episode source of execution affect its predictions.

At restored handoffs, that information can improve the continuation in both canonical generator families. Table 17 compares updated and reset state over the next committed segment on 24 tasks with eight outcome-blind anchors per task. Its lower panel adds the temporal control: from 64 common anchors, either observation-only or Stateful Flow generates the next prefix, with four noise draws per source (512 groups total). After executing that prefix, the stale reader retains the state from before the latest prefix, whereas the updated reader includes that prefix's realized transitions. Omitting this update lowers progress at the same receiving handoff. The interventions isolate which execution record the generator reads.

Table 17: **Current execution records support local continuation.** First-segment progress is the reduction in task-specific object–goal distance (m); contrasts have task-bootstrap 95% intervals. Each comparison holds the receiver handoff and model weights fixed. The upper panel shares noise. The lower panel reports Stateful-generated prefixes; updated, reset and stale readers share the next query's generator noise.

Policy	State contrast	Effect	95% CI
24 tasks, 192 anchors per family			
Transformer–Flow	Updated − reset	$+.004997$	$[.002096, .008559]$
U-Net–Diffusion	Updated − reset	$+.012102$	$[.003308, .022444]$
8 tasks, 64 anchors with Stateful-generated prefixes, four draws each			
Transformer–Flow	Updated − reset	$+.013041$	$[.006245, .020633]$
Transformer–Flow	Stale − updated	$-.010857$	$[-.019872, -.003614]$

The stale-record comparison shows why persistence requires updating: a record can retain earlier execution yet omit the physical response needed for the next continuation. Pooling both source policies gives updated-minus-reset progress of 13.84 mm and stale-minus-updated progress of -11.26 mm; the update remains useful when the preceding prefix comes from either generator. With this local utility established, exchanging real histories between receivers tests whether the useful adjustment is specific to the branch on which it acts.

F Branch-Specific Repair and Its Consequences through Requery

Execution history can guide a concrete repair: at reciprocal pick–place handoffs, each receiver's own record steers the same lifting movement in a different lateral direction and improves its return.

Crossed executions identify this history–action–value relation within the broader response across tasks. Separate multi-query comparisons examine how an initial mismatch develops under fresh feedback and how earlier actions change the states at which later decisions are made.

F.1 A HISTORY INTERVENTION MEASURES RECEIVER VALUE AND BRANCH ALIGNMENT

The comparison covers 50 tasks with eight receiver anchors per task in each canonical family. At a fixed receiver state, the native arm reads realized transitions accumulated from the episode start; the counterfactual arm reads its partner's history, matched by task, stage, and query index. Each partner also serves as a receiver, making the exchange reciprocal. The receiver observation, physical branch, and 16 candidate-noise identities remain unchanged. Each history generates its own bank, which is then held fixed for execution on the natural, mirrored, and replay branches. History therefore changes generation, while branch crossing measures the compatibility of the generated actions. Exchanging histories changes the stored reader state at $350/400$ anchors in each family.

Reciprocal pairs are drawn from expert trajectories within the same task, phase and query index. Matching minimizes a weighted distance between current state, recent state and the preceding executed action prefix, with weights 1, 0.5 and 0.25 after robust coordinate scaling. Selection first covers available phases, then fills four disjoint pairs per task by increasing distance. Each pair supplies two receivers; matching uses no candidate returns or episode outcomes.

Let $\bar{R}_b^a$ denote mean 16-step return over the bank generated with history $a \in \{\mathrm{native}, \mathrm{swap}\}$ and executed on branch $b \in \{+, -\}$, where $+$ is the receiver and $-$ its controlled counterbranch. The two contrasts are

$$\Delta_{\mathrm{native}} = \bar{R}_+^{\mathrm{native}} - \bar{R}_+^{\mathrm{swap}},$$
$$\Delta_{\mathrm{align}} = (\bar{R}_+^{\mathrm{native}} - \bar{R}_-^{\mathrm{native}}) - (\bar{R}_+^{\mathrm{swap}} - \bar{R}_-^{\mathrm{swap}}).$$

The first asks whether correct history improves receiver return; the second measures the history-bank–branch interaction. It is the same rectangular contrast used by the additive null in Section 2: with equal weight on the two branches and the two generated banks, diagonal pairing minus crossed pairing has value $\Delta_{\mathrm{align}}/2$. Its sign gives the favored pairing, while its magnitude measures sensitivity to that pairing. The swapped history comes from a reciprocal receiver, not from the edited counterbranch. Positive values of both contrasts identify the further property of useful repair: reading the receiver's own history improves its return and increases its advantage over the counterbranch.

F.2 RECIPROCAL ADJUSTMENT CLOSES THE HISTORY–ACTION–VALUE LINK

At the reciprocal pick–place pair, the two histories induce different lateral adjustments within the same lifting movement. Reading each receiver's own history raises its continuation value in both canonical families. Table 18 reports the receiver and alignment contrasts for the reciprocal pair alongside the task-balanced population means.

The native bank's return advantage is larger on its receiver than on the mirrored counterbranch. Both contrasts remain positive when the two histories exchange receiver roles, in both families. The `pick-place-v3` pair was identified within the completed comparison: $6/200$ Transformer–Flow pairs and $5/200$ U-Net–Diffusion pairs satisfy this reciprocal criterion, and the pick–place pair is the one shared by both families.

The action records make this repair concrete. Both histories produce a shared forward and upward movement, lifting the object from about 20 mm to 152 mm. Matching history chiefly adjusts the lateral component: relative to swapped history, the object's mean lateral endpoint shifts by $+2.07$ and -1.71 mm at the two receivers for Transformer–Flow, and by $+0.41$ and -0.33 mm for U-Net–Diffusion. Thus the history-dependent change reverses direction with the receiver, within an otherwise shared lifting motion. Receiver return improves for all 16 paired draws in each case. Its magnitude is uneven: Transformer–Flow's first receiver has a median improvement of 0.0468, compared with a mean of 0.1551. The case shows how a branch-compatible continuation can differ through a small adjustment to the same movement.

The signed response distinguishes repair from other uses of history. Each family has 14 of 50 task means positive on both contrasts, with seven tasks shared. At finer resolution, 40 of 168 shared

Table 18: **Branch-compatible repair within the full history response.** The upper panels give task-first mean 16-step return contrasts with task-bootstrap 95% intervals. The lower panel gives the two-receiver means for the shared reciprocal witness.

Population	Δ_{native} [95% CI]	Δ_{align} [95% CI]
All 50 tasks		
Pooled	$-.0193$ $[-.0946, .0466]$	$-.0095$ $[-.0546, .0340]$
Transformer–Flow	$+.0073$ $[-.0783, .0852]$	$-.0513$ $[-.1215, -.0018]$
U-Net–Diffusion	$-.0460$ $[-.1231, .0273]$	$+.0323$ $[-.0298, .1049]$
Low-recovery tasks, sustained interaction; two-family task mean	$+.3360$ $[.0286, .8413]$	$-.1240$ $[-.4312, .0438]$
Pick–place handoff; reciprocal branch-specific repair		
Transformer–Flow	$+.0879$	$+.0863$
U-Net–Diffusion	$+.00624$	$+.00610$

task–stage cells are both-positive for Transformer–Flow, 38 for U-Net–Diffusion, and 17 for both. These observed sign patterns map the variation across tasks and stages; the reciprocal pair resolves the motion adjustment at an individual handoff.

Task geometry alone does not locate such repairs. Among low-recovery tasks in sustained interaction, correct history improves mean receiver return, yet the mean alignment effect is negative. Nine of the 16 tasks in this subgroup have positive values of both contrasts. Likewise, task-level diagnostic severity is positively associated with receiver-return effects but not with alignment ($\rho = -.286$ for Transformer–Flow and $-.042$ for U-Net–Diffusion). These contrasts separate two properties of the reader. Its history dependence changes branch-relative value, and the reciprocal witness shows this change improving the receiving continuation. The negative Transformer–Flow alignment mean identifies an opposing preference elsewhere in the comparison. The reader therefore learns consequential branch-dependent adjustments; their direction determines whether the supplied history helps the receiver.

F.3 A HISTORY MISMATCH CAN GROW THROUGH LATER FEEDBACK

The local crossing identifies which history-generated actions fit a receiver. An independent multi-query experiment follows the effect of an initial history mismatch after the policy resumes receiving fresh feedback. Within one trained policy, we start two rollouts from the same query-2 snapshot. One reads its native execution state; the other reads a state from another trajectory of the same task, selected without using outcomes. Visible input and generator noise are initially shared. After this single swap, both receive fresh feedback and write only their own committed events under shared future noise. For each model, 50 tasks contribute eight outcome-blind state pairs and 16 noise draws per pair, giving 6,400 groups. The same groups are measured after $h \in \{1, 2, 4, 6\}$ future queries rather than resampled at each horizon. We write $\Delta_{6-1} = \Delta_{\text{native}-\text{swap}}(h = 6) - \Delta_{\text{native}-\text{swap}}(h = 1)$ for the primary temporal contrast. Table 19 compares all four backbone–generator cells; Figure 5 shows every horizon in the two canonical cells.

Both Flow backbones show a growing native-minus-swapped progress gap from one to six queries. The initial history mismatch therefore has a consequence that survives subsequent feedback and increases as execution proceeds. Diffusion yields a positive but uncertain change with U-Net and a near-zero change with Transformer. This comparison identifies recursive propagation in Flow and a different response to the same intervention in the tested Diffusion policies.

Retaining more history and amplifying a mismatch are distinct effects. The trained latest-prefix and Persistent policies also admit a temporal comparison. Both receive the latest realized prefix, but only Persistent keeps earlier commitments. At 50 tasks with eight expert handoffs and four noise draws per task–handoff, we measure how their object–goal progress difference changes between one and six queries. This contrast, A_{6-1}, is $-.003093$ $[-.008450, +.001973]$ for Flow and $+.004116$ $[-.002148, +.011516]$ for Diffusion (95% task-bootstrap intervals). Neither family resolves an increase. The Flow success advantage in Table 12 therefore establishes the value of earlier

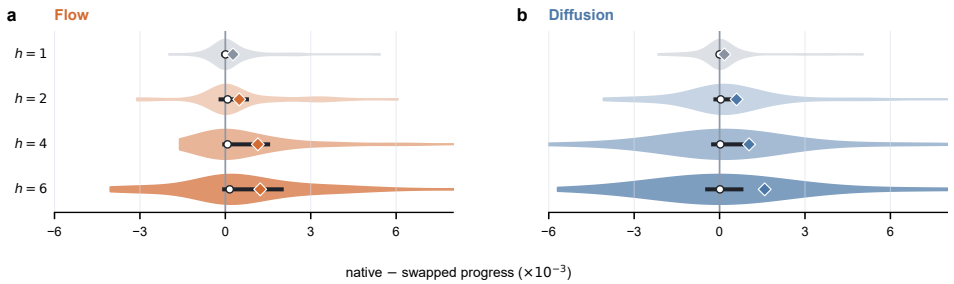


Figure 5: **Full recursive-pairing distributions in the canonical cells.** Task-level native-minus-swapped progress after one, two, four, and six future queries for Transformer–Flow and U-Net–Diffusion. Table 19 reports the $h = 6$-minus-$h = 1$ change. Diamonds are means, circles medians, and thick segments interquartile ranges.

Table 19: **Recursive separation is resolved for both Flow backbones.** Task-balanced native-minus-swapped progress on MT50; entries are scaled by 10^3. Each cell contains 6,400 paired groups at every horizon.

Backbone	Generator	$h = 1$	$h = 6$	Δ_{6-1} [95% CI]	Positive tasks
Transformer	Flow	.269	1.225	$+.956$ $[.114, 1.920]$	$30/50$
Transformer	Diffusion	$-.176$	$-.244$	$-.068$ $[-1.856, 1.582]$	$24/50$
U-Net	Flow	.174	2.216	$+2.042$ $[.498, 3.680]$	$32/50$
U-Net	Diffusion	.166	1.584	$+1.419$ $[-.213, 3.602]$	$25/50$

commitments without identifying it with growing short-horizon separation. The history-swap result above instead follows one initial disturbance within a fixed policy.

F.4 The handoffs a policy reaches shape its local effects

Execution also changes where the policy is asked to continue. Crossing the source of a handoff with its continuation policy isolates this contribution: the same continuation policy acts from handoffs reached by either the observation-only or execution-conditioned policy. On the query-2 comparison, Stateful Diffusion gains 6.375 success points at $h = 4$ when starting from its own previously reached states rather than those reached by the observation-only policy, with a task-bootstrap interval of $[.875, 12.128]$. Flow does not show the same direction.

This advantage depends on when the handoff is taken. The effect of changing the source policy changes sign across stages in $34/49$ eligible Flow tasks and $36/50$ Diffusion tasks. Earlier decisions thus alter not only which task state is reached but also whether that state is favorable for the continuation policy. This is the state-distribution dependence in the performance-difference identity: earlier actions change where later continuations are evaluated. Together, the local crossing and multi-query comparisons expose complementary ways in which execution can shape control, through the action chosen at a receiver and through the receivers reached next.

G Implementing the Execution-Conditioned Interface

Stateful Requery separates generation from commitment. The generator reads a fixed execution record while producing a chunk; the writer updates that record after the robot acts. Algorithm 1 generates $H = 32$ actions and commits at most $E = 16$. Each valid transition becomes one ordered writer event; termination shortens the sequence and padding acts as an identity update. The state starts at zero and changes only through committed events, so proposed but unexecuted actions cannot enter later context.

Algorithm 1 Stateful Requery: proposals read state; committed execution writes it

```
initialize execution state m ← 0
while the episode is active do
    O ← latest finite observation input
    freeze m; sample Y_{1:H} ~ π^G_θ(· | O, m)
    execute Y_{1:E} (or stop at termination)
    construct E from the committed state transitions
    m ← U_φ(m, E)
    discard the unexecuted suffix Y_{E+1:H}
end while
```

G.1 A FIXED-CAPACITY CARRIER STORES PREDICTION RESIDUALS

We use a KDA-style matrix state because it supports ordered, fixed-capacity updates over an arbitrary number of committed events and content-addressed reads from native generator features. Each event supplies normalized pre-transition state $\bar{s}_t$, normalized realized state velocity $\bar{v}_t$, and physical elapsed time Δt_t. For each head h, the writer forms a normalized key, a realized-motion value, continuous-time retention, and a write gate:

$$k_t^h = \mathrm{norm}(W_k^h \bar{s}_t), \qquad v_t^h = W_v^h \bar{v}_t, \tag{23}$$

$$\rho_t^h = \exp\left[-\Delta t_t \ \mathrm{softplus}(W_\rho^h \bar{s}_t + b_\rho^h)\right], \qquad \beta_t^h = \sigma\left(w_\beta^{h\top}[\bar{s}_t; \bar{v}_t] + b_\beta^h\right). \tag{24}$$

Here norm is per-head ℓ_2 normalization and σ is the logistic sigmoid. The retention ρ_t^h is a vector over key channels and β_t^h is one scalar per head. The retained matrix is revised by the component of realized motion that it did not already predict:

$$\widetilde{M}_t^h = \mathrm{Diag}(\rho_t^h) M_t^h, \tag{25}$$

$$\delta_t^h = v_t^h - (\widetilde{M}_t^h)^\top k_t^h, \tag{26}$$

$$M_{t+1}^h = \widetilde{M}_t^h + \beta_t^h k_t^h (\delta_t^h)^\top. \tag{27}$$

For Transformer token ℓ at query q, let $\widehat{z}_{q,\ell}$ be its layer-normalized feature. Each head reads the query's fixed matrix state, and a feature-dependent channel gate controls the residual:

$$u_{q,\ell}^h = \mathrm{norm}(W_u^h \widehat{z}_{q,\ell}), \quad c_{q,\ell} = \mathrm{concat}_h (M_q^h)^\top u_{q,\ell}^h, \quad \widetilde{z}_{q,\ell} = z_{q,\ell} + \sigma(W_g \widehat{z}_{q,\ell} + b_g) \odot W_c c_{q,\ell}. \tag{28}$$

The gate has the token's feature dimension. Projection notation suppresses affine biases; the U-Net condition-level read is described below.

The Transformer carrier has six action-mixer blocks. Each block owns six FP32 matrices of shape 64×64 per stream (576 KiB in total). The U-Net carrier owns one FP32 state with 32 heads of shape 64×64 (512 KiB). These per-stream states remain fixed in size as the event history grows and are separate from model parameters. Invalid suffix events set retention to one and the write gate to zero.

Total model sizes (observation-only / Stateful, in millions of parameters) are $34.00/37.60$ for Transformer and $265.70/282.07$ for U-Net on Meta-World; the corresponding LIBERO sizes are $34.03/37.65$ and $268.23/284.85$. These include the observation encoder and action generator and apply to both Flow and Diffusion within each backbone.

G.2 READ PATHS FOLLOW THE BACKBONE; SAMPLING FOLLOWS THE GENERATOR

The backbone determines the read path for both Flow and Diffusion. In the Transformer, each action token queries the persistent matrix after the block's within-query causal action mixer. The concatenated head output is linearly projected, multiplied by a learned sigmoid gate, and added to the token before cross-attention. This action-mixer state resets for every generator evaluation; the committed-execution matrices remain fixed throughout the chunk.

In the U-Net, the native observation condition queries the standalone matrix. The gated 512-dimensional result is concatenated with the native condition. Zero-initialized physical projections add this channel to every denoiser FiLM path without altering the native projection at initialization. The same condition is used throughout the chunk. Flow uses eight Euler steps; Diffusion uses 16 DDIM steps (Song et al., 2021).

G.3 TRAINING EXPOSES THE DEPLOYMENT CHRONOLOGY

A supervised query is one observation window paired with one target action chunk. Training uses the LeRobot Meta-World release (2,500 demonstrations, 204,806 frames) and the official LIBERO-90 release (4,500 demonstrations, 669,043 frames). Task-balanced sampling constructs queries within demonstration episodes. Observation windows repeat the first frame when necessary, and action loss masks positions beyond an episode's available targets. State and action coordinates use fixed training-set means and standard deviations; RGB values are scaled to $[0, 1]$. The Meta-World U-Net uses an ImageNet-initialized ResNet-18, while its Transformer and all LIBERO models initialize their visual encoders randomly. Each observation-only/Stateful pair shares its encoder initialization recipe and normalization statistics.

Execution-conditioned models train on four-query chronological segments with gaps of $\{8, 12, 16\}$ demonstration frames. Before each query, the writer scans only preceding realized transitions. State persists within an episode and resets at its boundary; detaching it between segments truncates backpropagation. Within a segment, gradients reach the writer, reader and generator. Supervision remains query-wise, conditioned on preceding execution rather than future events.

Optimization. We train the Flow and Diffusion policies with AdamW and a peak learning rate of 10^{-4}. A linear warmup is followed by cosine learning-rate decay. Table 20 summarizes the backbone-specific settings used on Meta-World MT50 and LIBERO-90.

Table 20: **Optimization settings for the generative policies.**

Parameter	Transformer	U-Net
AdamW (β_1, β_2)	$(0.9, 0.95)$	$(0.95, 0.999)$
Weight decay—action network	10^{-4}	10^{-6}
Weight decay—observation encoder	10^{-6}	10^{-6}
Warmup updates	1,000	500

For Transformers, biases, normalization parameters, and designated embedding parameters are excluded from weight decay.

On Meta-World, both Transformer interfaces train on 32 four-query segments per update; the observation-only model uses the same query targets without reading the execution record. The U-Net baseline samples 64 individual query targets, whereas Stateful U-Net uses 16 four-query segments. On LIBERO, every interface consumes the 128 targets from 32 four-query segments. Thus each trained pair has the same number of supervised queries per update and the same update budget. The four-query grouping is already included in these counts. The carrier comparisons in Table 2 use the corresponding Meta-World backbone's budget at seed 42.

Flow matching (Lipman et al., 2023) and Diffusion Policy (Chi et al., 2023) retain their native objectives. Flow draws a $\mathrm{Beta}(1.5, 1)$ variate, rescales it to $[.001, 1]$ for τ, and regresses the noise-minus-action velocity along a linear bridge. Diffusion predicts Gaussian noise at one of 100 timesteps.

The trained-control comparison uses the final 60,000-update Meta-World and 120,000-update LIBERO checkpoints at seeds 42, 43 and 44. Mechanism interventions keep the seed-42 models fixed: branch diagnosis uses the observation-only checkpoints, whereas reader, retention and recursive interventions use the execution-conditioned checkpoints. In the canonical Meta-World comparisons, these are Transformer–Flow and U-Net–Diffusion at 60,000 updates.

G.4 OBSERVATION INPUTS AND EVALUATION POPULATIONS

Meta-World uses all 50 manipulation tasks (Yu et al., 2020); LIBERO uses the 90-task pretraining suite (Liu et al., 2023). Both follow a single-view 128×128 RGB, proprioception, and one-hot task input: Meta-World uses `corner2` and 4-D end-effector/gripper state; LIBERO uses `agentview` and 8-D end-effector state, without eye-in-hand input. Meta-World rollouts allow 500 policy steps. LIBERO uses the official fixed initial states, five settling steps, and at most 400 policy steps.

Control and carrier comparisons use 50 rollout slots per task; other populations accompany their comparisons. Rollout contrasts pair task and slot, weight tasks equally, and bootstrap over tasks

within each training seed. Seeds provide independent replications. Fixed-noise interventions share generator innovation; old-suffix comparisons execute the previous chunk's remaining actions.

The full-task native results are shared across the trained-control and reader-reset tables. Execution-horizon comparisons use their stated 24-task population rather than the full-task aggregate. Reported intervals for the updated carrier and lifecycle comparisons use 20,000 task-bootstrap resamples.

G.5 The interface remains effective at finite generation budgets

An asynchronous controller executes available actions while generating the next chunk. On LIBERO-15 at 20 Hz, the success advantage persists with fewer generator evaluations (NFE; Table 21). Median latency falls below one control tick at small budgets; longer tails motivate asynchronous execution.

Table 21: **Execution conditioning remains useful at small generation budgets.** Asynchronous LIBERO-15 at 20 Hz. Gains compare execution-conditioned Flow with its reset-state execution.

NFE	Median latency (ms)	95th percentile (ms)	Success gain (pp)
1	12.7	58.3	$+53.07$
2	21.3	114.8	$+54.40$
4	41.5	232.7	$+55.33$
8	84.3	442.9	$+48.00$

The benefit also survives a different execution schedule: blocking LIBERO-15 at NFE1 retains $+48.13$ points over observation-only Flow.