NEEDLEWORK: OFFLINE REWRITING OF ROBOT DATA WITH VERIFIED LOCAL STITCHES

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot demonstrations may contain useful behavior even when individual episodes are inefficient or unsuccessful. Trajectory stitching offers a way to compose these behaviors into improved training data, but identifying useful connections and verifying their feasibility is difficult in high-dimensional robot data, where many prior methods rely on low-dimensional state representations. We introduce NEEDLE, an offline dataset-augmentation algorithm that addresses these challenges by adding short, verified action bridges between recorded observations in high-dimensional robot demonstrations. First, NEEDLE identifies and creates connections that bypass suboptimal detours, broaden action coverage, and augment the original dataset with failed trajectories, using only RGB images, proprioception, and episode-level outcomes, without new environment interaction or privileged object state. Next, we present a sampling technique that incorporates accepted bridges into policy training without synthesizing intermediate images or discarding the original demonstrations, allowing policies to learn alternative actions while retaining the original dataset's coverage. On real-robot tasks, NEEDLE improves success rate over the strongest baseline on each task by an average of 21 percentage points. Videos and supplementary materials are on needle-work.github.io.

1 INTRODUCTION

The best robot behavior may not exist in any single trajectory of a dataset. Consider a bimanual laundry-folding task. One demonstration may fold the sleeves reliably but require several corrective grasps to complete the bottom fold; another may fold the sleeves poorly but fold the bottom cleanly. Neither demonstration alone completes the task well, but synthesizing a connection between the two good segments would produce a demonstration better than either individually.

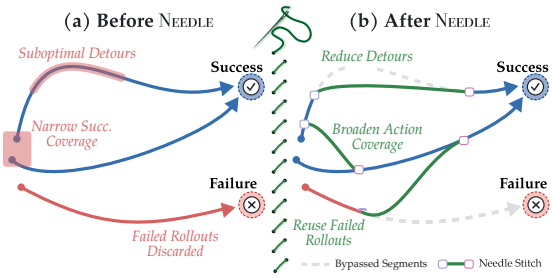

Figure 1: NEEDLE produces synthetic bridges within trajectories that bypass inefficient detours, cross-trajectory stitches that discover more successful ways to complete the task, and recover useful behavior from failed episodes.

This example illustrates one of three opportunities offered by **trajectory stitching** (Fig. 1). *Within a trajectory*, it can bypass inefficient detours within a demonstration, allowing a policy to learn shorter routes to success. *Across successful trajectories*, it can combine complementary behaviors and expose continuations that were never recorded together. *From failure trajectories*, it can recover useful behavior from failed episodes by connecting states along an unsuccessful execution to a successful continuation.

Despite this promise, realizing this in real-world robot manipulation is difficult. A stitch must be both *useful*, offering a better continuation, and *feasible*, with actions that physically reach its target. Most prior methods that stitch trajectories typically assume access to low-dimensional or privileged state (Yu et al., 2025; Lee & Choi, 2025; Hepburn & Montana, 2024; Li et al., 2024; Char et al., 2021), while methods designed to improve mixed-quality robot data typically filter or reweight

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.

recorded behavior without supplying the actions needed to connect desirable segments (Du et al., 2023; Hejna et al., 2025; Agia et al., 2025; Dass et al., 2026). We ask if we can get the best of both.

We introduce **N**ew **E**dges from **E**xisting **D**emonstrations through **L**ocal **E**diting (NEEDLE), which identifies valuable segments with no existing connection, and proposes verified action bridges to connect them. Our key insight is that **rather than learning a globally accurate dynamics model or task-specific value function and asking either to reason over long horizons, we can reduce stitching in high-dimensional spaces to a series of locally supervised reachability problems.**

Specifically, the feasibility and duration of a local connection can be learned directly from recorded action sequences. Across any episode, successful, suboptimal, or failed, each short segment is an example of which two observations can be connected, the specific action sequence that connects them, and the exact number of steps it takes. We use this insight to train two modules. Given source and target observations, a goal-conditioned diffusion Inverse Dynamics Model (IDM) generates action sequences that attempt to reach from source to target. In parallel, we supervise an action-conditioned verifier that receives the same source and target observations and predicts whether, and after how many steps, the proposed actions reach the target.

To make these connections useful for policy learning, NEEDLE turns accepted bridges into action sequences at recorded observations. Pairing each bridge with the originally-recorded actions leading up to its source exposes the synthesized actions from multiple preceding observations, without needing to generate intermediate images. Original training examples remain available, including those the bridge bypasses, so the policy receives new ways to make progress without losing supervision for the situations and behaviors covered by the original demonstrations.

Our contributions are threefold:

(1) An offline stitching method that finds useful connections and verifies bridges with an action-conditioned reachability model.

(2) A training formulation that adds bridge actions while preserving recorded supervision, without synthesizing observations or requiring a shared coordinate frame.

(3) Evaluations on a real robot and in simulation of success, completion efficiency, recovery, and key design decisions.

2 RELATED WORK

Explicit Stitching of Offline Datasets. Explicit dataset stitching materializes a connection between recorded segments of a trajectory as new training data. Existing methods select connections using signals such as task value, return, temporal structure, or coverage (Char et al., 2021; Hepburn & Montana, 2024; Li et al., 2024; Lee & Choi, 2025; Yu et al., 2025; Chen et al., 2026a). They assess connectivity through learned dynamics, successor likelihoods, or proximity in a learned temporal metric (Char et al., 2021; Hepburn & Montana, 2024; Li et al., 2024; Lee & Choi, 2025; Yu et al., 2025). Generative approaches additionally synthesize states or longer trajectories for policy training (Li et al., 2024; Zhou et al., 2024; Lee & Choi, 2025; Yu et al., 2025; Chen et al., 2026a). NEEDLE instead formulates multistep connectivity as action-conditioned reachability between recorded observations, adding action supervision without reconstructing intermediate images. Connection utility depends on continuation lengths and success rather than a learned value function.

Curation via Filtering, Retrieval, and Reweighting. Filtering, retrieval, and reweighting methods determine which recorded behavior should influence policy learning (Du et al., 2023; Wu et al., 2025a; Hejna et al., 2025; Chen et al., 2025; Agia et al., 2025; Dass et al., 2026). They vary in granularity, from complete demonstrations to subtrajectories or individual transitions, and use signals such as relevance, progress, graph-based value, or backward reachability to identify useful behavior (Memmel et al., 2025; Yin & Abbeel, 2024; Yue et al., 2024; Wang et al., 2025; Chen et al., 2026b). These approaches can recover useful supervision from heterogeneous data while leaving the recorded action targets unchanged. More broadly, reusing recorded trajectories to construct corrective supervision has been explored in learned models (Venkatraman et al., 2015) and policies (Ke et al., 2024; Zhang et al., 2024). NEEDLE similarly creates supervision from existing trajectories, but constructs alternative action sequences connecting recorded observations, rather than corrections for predicted states.

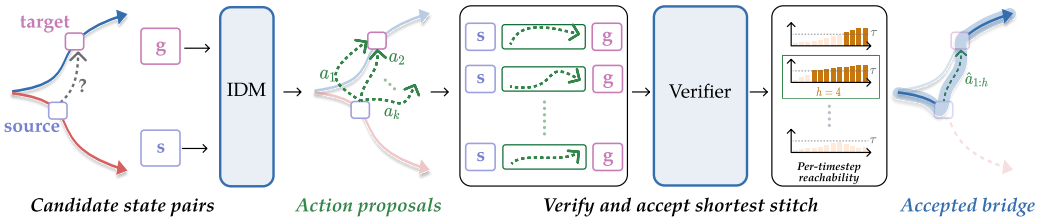

Figure 2: **From candidate connection to verified bridge.** Given a source-target pair (s, g), the inverse dynamics model (IDM) samples K action proposals. The action-conditioned verifier predicts whether and when each proposal reaches g. We select the shortest accepted prefix as a bridge.

Implicit Stitching and Compositional Planning. Offline RL can improve on recorded behavior by propagating value across transitions and composing favorable continuations through policy learning (Kumar et al., 2022; Singh et al., 2021; Chebotar et al., 2023; Park et al., 2023; Hansen-Estruch et al., 2023). Goal-conditioned formulations also relate action selection to learned estimates of future reachability (Eysenbach et al., 2022; Kaelbling, 1993). Generative planners construct trajectories or subgoals at inference (Janner et al., 2022; Ajay et al., 2023; Black et al., 2024; Huang et al., 2024; Kim et al., 2024; Luo et al., 2025), while graph-based approaches search over recorded states using learned goal-reaching relationships (Eysenbach et al., 2019; Emmons et al., 2020). Related policy-steering methods search over or evaluate action proposals using predicted outcomes at deployment (Jain et al., 2025; Wu et al., 2025b; Ren et al., 2024). In these approaches, composition is expressed through learned values and policies, or inference-time planning and steering. NEEDLE materializes its connections offline as explicit action supervision, so the resulting imitation policy requires neither connection search nor verification at deployment. We select baselines from each category to evaluate against in Section 4.

3 How Needlework Works

NEEDLE answers two separate questions when rewriting a dataset: whether a stitch is *useful*, and whether a stitch is *feasible*. Candidate selection addresses the first question. An Inverse Dynamics Model (IDM) and an action-conditioned verifier address the second. We formalize the notation in §3.1, the selection of useful connections in §3.2, action generation and verification in §3.3, and policy training on the rewritten dataset in §3.4. Additional details in Appendix D.

3.1 Notation

Let $\mathcal{D} = \{(\tau^i, y_i)\}_{i=1:N}$ be a dataset of N trajectories $\tau^i = (o_1^i, a_1^i, \ldots, a_{T_i-1}^i, o_{T_i}^i)$, where y_i indicates whether trajectory i succeeds or fails. Each observation $o_t^i = (x_t^i, p_t^i)$ contains multi-view RGB images x_t^i and robot proprioception p_t^i. We use i, j to index trajectories and t, t' to index timesteps within them. A stitch connects a source observation $s = o_t^i$ to a target observation $g = o_{t'}^j$ through a newly generated action sequence. We use *proposal* for a length-H action sequence $\hat{a}_{1:H}$ sampled from the IDM, *bridge* for an accepted prefix $b = (s, g, \hat{a}_{1:\ell})$, where $\ell \leq H$, and *stitching* for the operation that incorporates a bridge into the policy-training data.

3.2 Where should we stitch?

A feasible connection is not necessarily useful. Connecting nearby observations may simply reproduce existing behavior, or worse, introduce an unnecessary detour. We therefore search for source-target pairs whose recorded continuations offer an opportunity to improve the dataset.

We define three sources of candidate pairs, with different notions of "improvement" as illustrated in Figure 1, and retrieve candidates from these categories before proposing actions. A *within-trajectory shortcut* (S1) bypasses a longer segment of a successful trajectory. A *cross-trajectory transplant* (S2) connects a successful source to a shorter successful continuation in another trajectory. A *failure funnel* (S3) connects a source on an unsuccessful trajectory to an observation on a successful trajectory. Appendix B.2 isolates the effect when training on different subsets of these three stages.

Defining a good stitch. For a source $s = o_t^i$ and target $g = o_{t'}^j$ on successful trajectories, the recorded source continuation requires $T_i - t$ steps, while the target has $T_j - t'$ steps remaining. If

an ℓ-step bridge reaches g, its potential reduction in completion steps is

$$\Delta_{\text{cont}}(b) = (T_i - t) - [\ell + (T_j - t')]\,. \tag{1}$$

We retain successful-to-successful bridges only when $\Delta_{\text{cont}}(b) > 0$. This criterion favors connections whose shorter remaining path outweighs the cost of reaching the target, conditional on the bridge succeeding and the target's continuation remaining applicable.

For failed sources, there is no successful continuation against which to measure steps saved. We instead seek connections to successful trajectories, preferring the shortest combined bridge and remaining successful continuation. These bridges introduce action supervision toward success from states that would otherwise contribute no successful continuation.[1]

We retain at most one bridge from each source, selecting the accepted bridge with the shortest combined bridge and successful suffix, $\ell+(T_j-t')$. For successful sources, minimizing this quantity also maximizes the estimated continuation gain Δ_{cont}, and we require that gain to be positive. The same bridge-plus-suffix criterion ranks failure funnels. We discuss how to leverage these bridges for training in the following Section 3.4.

3.3 Training the Needle and Thimble

A useful connection improves the dataset only if its proposed actions reach the intended target. We address this problem by individually training an IDM and verifier, thus separating the challenges of action generation and verification (Swamy et al., 2026). At a local timescale, **each segment provides examples of *which* observations can be connected, and *how***. We leverage this insight to design positive supervision from short-horizon segments within both successful and failure trajectories, thus expanding the amount of usable training data for our IDM and verifier.

Goal-conditioned action proposals. Following the goal-conditioned inverse-model formulation in Pathak et al. (2018), we instantiate the IDM as a goal-conditioned diffusion model

$$\hat{a}_{1:H} \sim \pi_\theta\left(\cdot \mid s, g\right), \tag{2}$$

trained with the standard diffusion denoising objective (Ho et al., 2020; Chi et al., 2023). Training examples come from demonstrated segments known to be reachable within the action horizon H. For a source $s = o_t^i$, we sample an offset $1 \le \delta \le H$, set $g = o_{t+\delta}^i$, and supervise the model with recorded actions up to that goal. If $\delta < H$, we hold the last action at the goal pose to fill the action horizon, thereby also encouraging greater sensitivity to the specific action that reaches the goal. At stitching time, the trained IDM samples K action proposals for each candidate pair.

Action-conditioned verification. A goal-conditioned IDM produces plausible actions; the verifier needs to certify *whether* the goal is reached by this set of actions, and *how early*.[2] Thus, the verifier receives a source, goal, and proposed action sequence and predicts whether the goal has been reached by each prefix:

$$(r_1, \ldots, r_H) = f_\phi(s, g, a_{1:H}), \tag{3}$$

where r_h scores whether executing $a_{1:h}$ from s reaches g at or before step h. Causal attention over the action tokens prevents r_h from depending on actions after step h. The verifier and IDM share a visual encoder for the source and goal observations, and can be trained in parallel.

Predicting reachability at multiple action-prefix lengths lets us select both a connection and its duration, which is needed to evaluate its contribution to steps to completion (ℓ in Eq 1). For recorded short segments, the actions and their designated target observations provide direct supervision for local reachability. We can therefore assess whether an action sequence connects two observations without estimating long-horizon returns or reconstructing intermediate states.

We construct three categories with the same anchor for training. Formally, let o_t^i be the shared source observation, and $a_{t:t+\delta^+}$ the recorded action chunk from o_t^i, with $\delta^+ \le H$. Let A_t^i denote this chunk padded to length H by holding the goal pose when $\delta^+ < H$, as described for the IDM above.

[1] Enumerating all source-target pairs would be expensive. In practice, we use the IDM's visual encoder features to retrieve a finite neighborhood of candidate targets for each source. Retrieval identifies promising endpoints but does not establish that they can be connected.

[2] A "thimble", if you will, preventing harm from bad stitches.

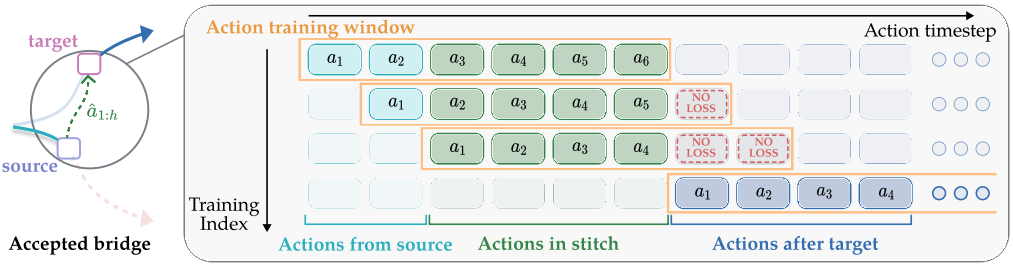

Figure 3: **Training on a bridge without intermediate observations.** Each example pairs a recorded observation with the original actions leading to the bridge (cyan), followed by the generated bridge actions (green). The loss stops when the bridge ends (red). Original demonstration windows remain available as separate training examples (blue).

We sample a single positive *within-horizon goal* $g^+ = o^i_{t+\delta^+}$, whose binary target is positive for every $h \geq \delta^+$ and negative beforehand. We also pair with a *beyond-horizon goal* $g^{\text{far}} = o^i_{t+\delta^{\text{far}}}$, where $\delta^{\text{far}} > H$, and with a *cross-trajectory goal* $g^\times = o^j_{t'}$, where $j \neq i$. These latter two categories are negative pairs to the first positive anchor. Note that all pairs are conditioned on the same action chunk A^i_t. In particular, labeling our cross-trajectory tuple $\left(o^i_t, g^\times, A^i_t\right)$ negative does not mean that $\left(o^i_t, g^\times, A'\right)$ is negative for every possible A'. In fact, finding an alternative action chunk that connects such cross-trajectory observations is the objective of stitching.

We train all prefix outputs with binary cross-entropy. Since false-positive predictions introduce incorrect action supervision, we select a threshold τ_h for each horizon on held-out trajectories, prioritizing a low empirical false-positive rate (Tong et al., 2018). We consider a prefix supported when its score exceeds the threshold used for its stitch stage.

3.4 Training a Policy from Stitches

Each policy-training example pairs recorded observations with a short sequence of actions to learn. A bridge supplies new actions but no images along the way. Its endpoint may also differ slightly from the recorded target, and the target demonstration may express its actions in a different coordinate frame, as in UMI (Chi et al., 2024; UMI Robot Dataset Community, 2026; Patel et al., 2026). We account for all three limitations when constructing training examples (Figure 3).

Consider an accepted bridge $b = (o^i_t, g, \hat{a}_{1:\ell})$. Rather than constructing a single training example at its source, we also construct examples beginning at preceding recorded observations. For a window starting q steps before the bridge, the action target is

$$A^{(q)} = \left(a^i_{t-q}, \ldots, a^i_{t-1}, \hat{a}_1, \ldots, \hat{a}_\ell\right). \tag{4}$$

Varying the starting point exposes the same connection through multiple training windows, allowing the policy to learn the new continuation from several preceding observations. Further, if a bridge ends before the policy's fixed action prediction horizon, we mask all action positions after its endpoint to prevent the policy from being trained on an artificial concatenation of the bridge and the target demonstration. Even when a bridge approximately reaches its target, the recorded actions following that target may not be valid from the bridge's actual endpoint. We instead learn the target continuation separately, paired with its own recorded observations.

Preserving original coverage. Because a stitch creates an alternative local continuation rather than an exact synthetic trajectory, we still keep the original actions at bridge sources and along the segments they bypass, and control their sampling frequency relative to the new bridge windows. Our training construction preserves the original supervision and stops the new action target at the bridge endpoint. App. B.3 shows that removing these safeguards generally reduces downstream success.

4 Experiments

We evaluate whether rewriting an offline robot dataset with NEEDLE improves downstream policy performance, and, more importantly, *how* the rewritten data changes the behavior learned by the pol-

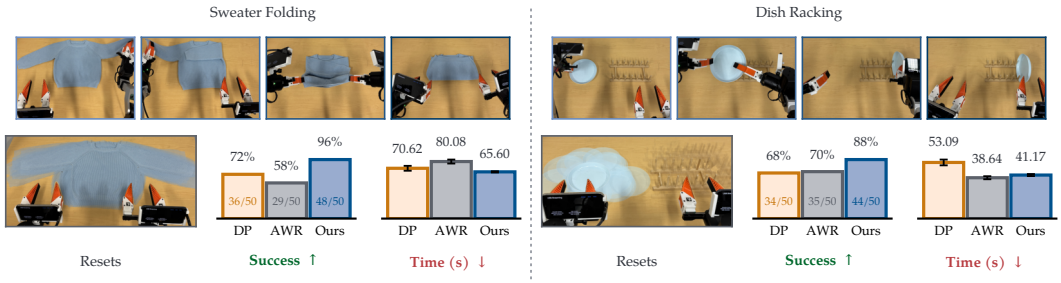

Figure 4: **Real-robot tasks and evaluation.** We evaluate NEEDLE on two real-robot bimanual tasks: Sweater Folding and Dish Racking. We show task sequences, reset distributions, success rates, and completion times. Each policy is evaluated 50 times per task at paired starting configurations across all methods. Completion times show mean and standard error over successful trials.

icy. After describing the experimental setup (§4.1), we first measure task success on real-robot and simulated manipulation tasks (§4.2). Then, we measure completion within a fixed budget on both real-robot and simulated tasks, and use controlled simulation experiments to isolate two effects of stitching: reaching useful states sooner and extending supervision to states encountered in failed trajectories (§4.3). Finally, we test whether verification is necessary for selecting useful bridges (§4.4) and examine NEEDLE's performance when the demonstrations are already near-optimal (§4.5).

4.1 EXPERIMENTAL SETUP

Tasks and data. We evaluate sweater folding and dish racking on a real robot, using demonstrations collected with handheld UMI devices (Chi et al., 2024; Patel et al., 2026); policies receive wrist-camera RGB and proprioception measured relative to each trajectory's starting position (Figure 4). In simulation, we use the suboptimal Robomimic Multi-Human *Can*, *Square*, and *Transport* tasks (Mandlekar et al., 2022), with the same 40 successful demonstrations for every method and 150 additional failures for methods that use them. We leverage simulation because it permits controlled resets that let us compare continuations from identical states. We also test three DexMimicGen tasks with near-optimal demonstrations (Jiang et al., 2025). All methods use RGB, proprioception, and episode outcomes, without per-step rewards, privileged object state, or new environment interaction.

Baselines. We compare against an explicit stitching baseline **MBTS** (Hepburn & Montana, 2024), selection and reweighting baselines **SBR** (Wang et al., 2025), **AWR** (Peng et al., 2019; Kostrikov et al., 2022), and **SARM** (Chen et al., 2026b)), and an implicit stitching baseline **IDQL** (Hansen-Estruch et al., 2023). We adapt SARM to the same level of supervision as other baselines, without the annotated subtask boundaries or language labels from its released implementation. IDQL reranks diffusion-policy samples with the IQL critic also used by AWR. Across tasks, all methods use the same DINOv3 vision encoder (Siméoni et al., 2025) and Diffusion Policy architecture (Chi et al., 2023), so comparisons isolate differences in training supervision. App. A provides more information on evaluation details and task setups; App. C shows more detail on baseline and our tuning efforts.

4.2 DOES NEEDLE IMPROVE POLICY SUCCESS?

Rewriting improves real-robot success. NEEDLE improves success by 24 percentage points on sweater folding and 18 points on dish racking over the strongest baseline (Figure 4). We examine which stages account for these gains by reviewing all 50 trials per policy and task.

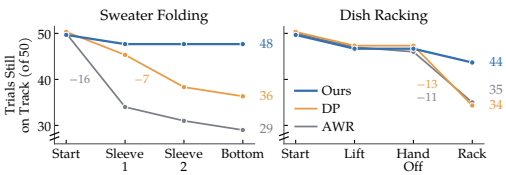

Figure 5: **Fewer failures across task stages.** Trials completing each successive stage, out of 50 per policy and task, from video review. Endpoint labels give final success counts, and the negative numbers mark each baseline's largest loss of trials at a single stage.

We seek to localize where and how imperfect demonstrations affect baselines. In Figure 5, we decompose where each checkpoint fails over the course of a rollout. For both baseline policies, their main failure mode is repeating a suboptimal motion from demonstrations cyclically, such as folding and unfolding a sleeve, or circling its gripper around the rack for prolonged time before

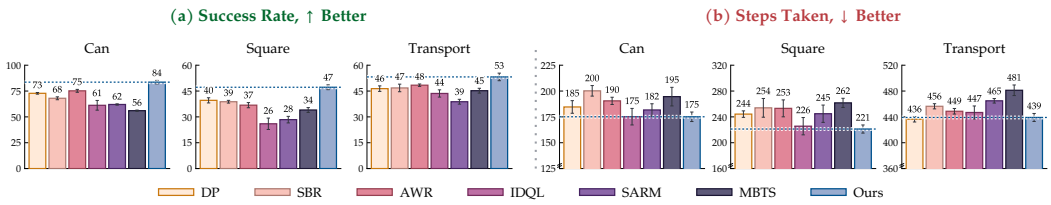

Figure 6: **Policy performance on suboptimal Robomimic demonstrations. (a)** NEEDLE achieves higher average success than all baselines. **(b)** Its successful rollouts take fewer steps on average than DP's on *Can* and *Square*. Bars show the mean and standard error over five training seeds, with 50 held-out initial conditions per seed; completion steps are averaged over successful trials.

placing imprecisely. Appendix B.1 complements this analysis, showing NEEDLE having the lowest percentage of cyclic motion repeats across the two tasks.

The same advantage appears with suboptimal simulated demonstrations. On the three Robomimic tasks, NEEDLE achieves higher average success than every evaluated baseline (Figure 6). Averaged across tasks, rewriting improves success by 8.4 percentage points over standard DP and by 6.9 points over the strongest competing baseline on each task. Though these aggregate success rates establish that rewriting helps, they do not by themselves explain why. To identify which parts of the design contribute to these gains, Appendix B.2 isolates the stitching stages, Appendix B.3 ablates different training variations with the stitched dataset, and Appendix B.4 tests whether the gain comes from the bridge actions themselves. We next use the controlled resets available in simulation to examine what the accepted stitches actually change.

4.3 WHAT DOES NEEDLE CHANGE ABOUT THE LEARNED BEHAVIOR?

We first measure whether rewriting improves completion within a fixed budget on real-robot and simulated tasks. We then use controlled simulation resets to test whether accepted bridges shorten the remaining path to success and whether failure funnels improve recovery from states outside the successful-demonstration distribution.

Rewriting yields more success within a fixed budget. Mean completion time among successes can be misleading: a policy that solves fewer difficult starts may appear faster. We therefore measure the fraction of all evaluation trials completed within different time-budget thresholds, with failures counted against it, shown in Figure 7. On both real-robot tasks, NEEDLE reaches comparable success sooner and ultimately completes more trials than either baseline. The same pattern holds in simulation, shown in Figure 8, right. Evaluated from matched initial conditions, NEEDLE reaches comparable success in fewer steps and solves more starts by the full horizon.

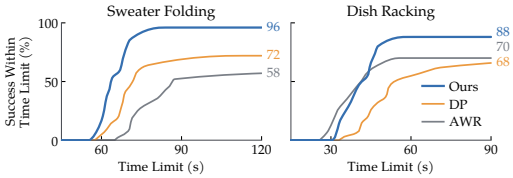

Figure 7: **NEEDLE finishes more trials within a fixed budget.** Labels give final success rates per time limit over the 50 paired trials.

Accepted bridges move the policy toward shorter suffixes. From each state that NEEDLE finds a valid stitch for, we execute the stitch's full action sequence and the original continuation for the same number of steps, then resume the same DP from the two resulting states in simulation. This can be viewed as an advantage-style comparison under DP: the two continuations begin from the same source, and we ask whether our stitches leave the policy in a state from which success is quicker to achieve. The left panels of Figure 8 show that, across all three tasks, accepted bridges shift the distribution toward fewer remaining steps to success, confirming the behavior that the continuation-gain criterion from Equation 1 seeks to induce.

Failure funnels add supervision outside the successful-demonstration distribution. The second effect is qualitatively different. A failed trajectory may contain useful intermediate behavior even though it never reaches the task goal. A failure funnel connects such a state to a successful continuation, converting a state that previously had no successful action target into one with explicit supervision toward success. The left panels of Figure 9 visualize this effect. Starts introduced by

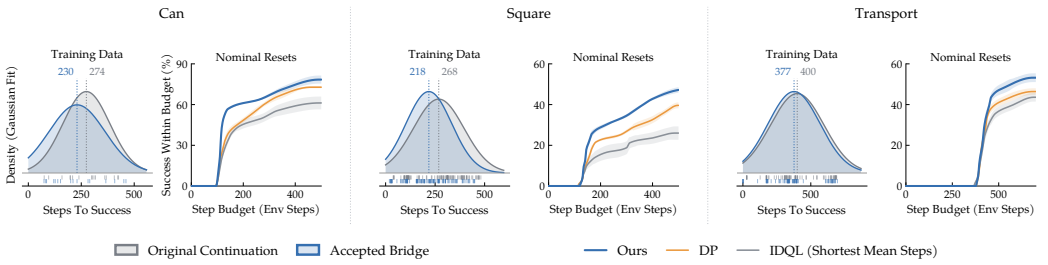

Figure 8: **Shorter continuations and higher success within a step budget.** For each task, the left panel shows remaining completion steps after executing either the original continuation or an accepted bridge, restricted to pairs where both continuations succeed. On average, bridges lead to shorter remaining continuations. The right panel shows the fraction of matched nominal starts solved within each budget. NEEDLE reaches comparable success sooner and solves more starts by the full horizon. We plot mean and S.E. for five training seeds, each with 50 held-out starts.

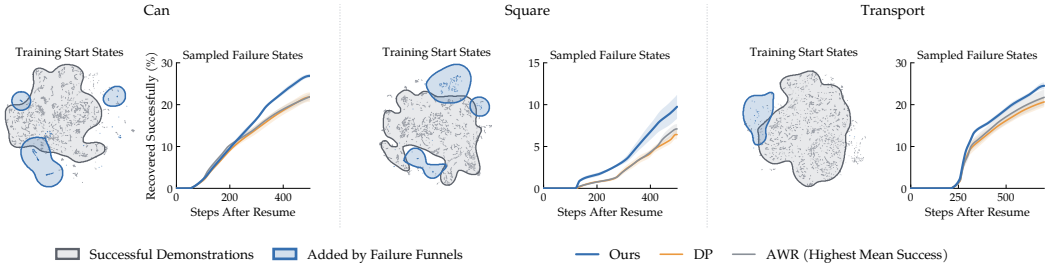

Figure 9: **Expanded training coverage and recovery from failure states.** For each task, the left panel shows successful-demonstration states and starts added by failure funnels, visualized as 90% density regions in a t-SNE embedding of the visual encoder. The right panel measures recovery after resuming from OOD states, where NEEDLE recovers more often than DP and AWR, the non-DP baseline with the highest average success rate across all three tasks on nominal resets. Curves show mean and standard error for five training seeds, each with 50 held-out starts.

failure funnels occupy regions of the learned visual representation that extend beyond those covered by successful demonstrations.

Additional coverage from failure improves recovery. We test whether this broader supervision changes closed-loop behavior by resetting each policy to the same randomly-sampled failure states and measuring subsequent recovery. NEEDLE recovers more often than DP on all three evaluated populations (right panels of Figure 9) and also outperforms the strongest non-DP baseline, AWR.

4.4 DOES VERIFYING FEASIBILITY MATTER?

A useful connection must meet two separate tests. It must be useful: following the target's recorded actions should give the policy a better route to task completion. It must also be feasible: the proposed bridge actions must reach the target. Candidate selection estimates utility from recorded continuations; here we test whether the verifier predicts feasibility from the proposed actions, whether that prediction reflects physical bridge quality, and whether filtering on it improves downstream learning.

The verifier distinguishes actions and selects closer landings. We first ask how sensitive the verifier is to the proposed action, as opposed to just relying on the source and goal inputs. For each held-out source–goal pair, we replace the demonstrated action chunk with a logged chunk from a distant time in the same episode or from another episode, selected randomly, at the same relative episode phase, or by nearest starting arm pose. Figure 10(a) shows that the verifier distinguishes the demonstrated actions across all substitute types (AUROC 0.85–0.98), including the closest-pose substitutes on each task.

Next, we replay proposed bridges in simulation and measure the larger of the arm and object position errors relative to the target (Figure 10(b)); this error is available only in simulation and is never observed by NEEDLE. Accepted bridges are more likely than rejected bridges to finish within

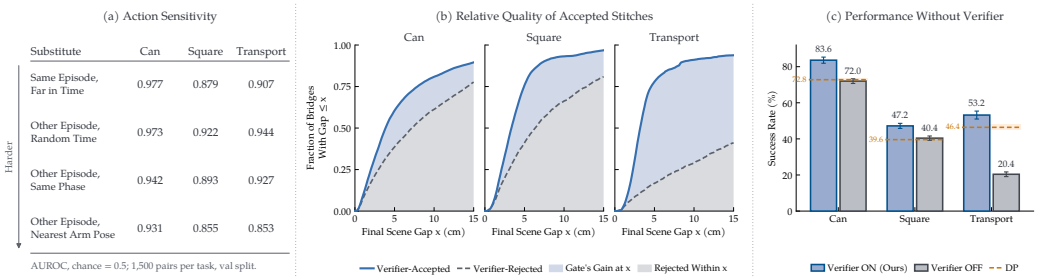

Figure 10: **Verifier-based selection connects action sensitivity, bridge quality, and policy success.** **(a)** With the source and target fixed, the verifier distinguishes demonstrated actions from logged substitutes on 1,500 held-out pairs per task. **(b)** Accepted proposals more often land near the target than rejected proposals. Scene gap is the larger of the arm and object position errors, measured only for this diagnostic; NEEDLE itself never observes object positions. **(c)** Removing verifier filtering reduces downstream success on all three tasks with the IDM and bridge counts per stage held fixed. Bars show success and standard error over five training seeds of 50 evaluations each.

5 cm on all three tasks, by 21.9, 34.6, and 61.1 percentage points on *Can*, *Square*, and *Transport*, respectively. In absolute terms, about 61%, 76%, and 78% of accepted bridges land within 5 cm respectively on these three tasks. These results show that the verifier responds to the action sequence and selects bridges that more often land near the target.

Verifier-based selection improves the downstream policy. To test whether this selection matters for learning, we compare performance against policies trained on just the IDM proposals, while keeping the number of training bridges per stage the same as what was accepted by the verifier. The resulting policy loses success on all three tasks (Figure 10(c)); on *Transport*, success falls from 53.2% to 20.4%, a decrease of 32.8 percentage points (61.7% relative). Thus, the added data's value depends on which bridges are selected, beyond the number of examples added.

4.5 WHAT IF EXISTING DEMONSTRATIONS ARE ALREADY NEAR-OPTIMAL?

Table 1 reports how much rewriting helps when demonstrations are already near-optimal, and highlights an important boundary condition. NEEDLE's percentage gain is smaller, at a maximum gain of four percentage points. Thus, NEEDLE is most valuable when an offline dataset contains complementary, inefficient, or unsuccessful behavior that can be recomposed into better supervision. When the demonstrations are already close to the desired behavior, rewriting has correspondingly less to add; nevertheless, it does not reduce success relative to standard BC.

Task	DP	**Ours**
Coffee	62.0 ±1.2	**63.3** ±0.7
Three Piece Assembly	51.3 ±0.7	**55.3** ±1.8
Threading	**46.0** ±1.2	**46.0** ±1.2

Table 1: **Policy success with near-optimal DexMimicGen demonstrations.** NEEDLE matches or modestly improves on DP across the three tasks. Values are success rates (%), reported as mean and standard error over three training seeds with 50 held-out initial conditions per seed.

5 CONCLUSION

We present NEEDLE, an offline dataset-augmentation method for composing useful behavior within a dataset of imperfect robot demonstrations. NEEDLE reduces stitching to short, locally supervised reachability problems. Across real-robot and simulated manipulation tasks, our results show that this formulation can turn inefficient, complementary, and unsuccessful behavior into useful new supervision.

Limitations. NEEDLE assumes that useful behavioral composition can be realized through short local action bridges. It may therefore fail when reaching a candidate goal requires long-horizon contact dynamics, substantial object reconfiguration, or feedback unavailable within an open-loop action chunk. Its verifier is statistically calibrated rather than a formal feasibility or safety guarantee, so false-positive bridges can introduce harmful supervision.

AI USE STATEMENT

In this work, we used generative AI tools to integrate baselines from publicly released code into our training pipeline, support qualitative data analysis, and to obtain feedback on the manuscript, including critiques of experimental design, baseline choices, and analysis in the form of simulated peer review. The authors designed and ran all experiments and decided which feedback to act on. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims or proofs, propose or refine hypotheses, assist with translation, or clean or reformat datasets. Additionally, we used generative AI tools to format the supplementary website, identify and summarize relevant literature, format references, and clean up our development codebase for public release. We reviewed all AI-assisted work, and all AI-suggested text edits were reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix A documents dataset collection, composition, and selection, task setups, and checkpoint selection. Appendix D provides model architectures, optimization settings, hyperparameters, verifier calibration, stitching and training-window construction, and compute requirements. Appendix C details baseline adaptations and tuning. Evaluation and diagnostic protocols are described in Appendix D.6. Additional results and ablations are provided in Appendix B. We will open-source all the code and algorithms presented in this paper, from training to dataset rewriting.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. CUPID: Curating data your robot loves with influence functions. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2907–2932, 2025.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations*, 2023.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *International Conference on Learning Representations*, 2024.

Ian Char, Viraj Mehta, Adam Villaflor, John M. Dolan, and Jeff Schneider. BATS: Best action trajectory stitching. In *NeurIPS Workshop on Offline Reinforcement Learning*, 2021.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Anand Sontakke, Grecia Salazar, Huong T. Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singh, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn, and Sergey Levine. Q-Transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3909–3928. PMLR, 2023.

Annie S. Chen, Alec M. Lessing, Yuejiang Liu, and Chelsea Finn. Curating demonstrations using online experience. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.071.

Chang Chen, Hany Hamed, Doojin Baek, Taegu Kang, Samyeul Noh, Yoshua Bengio, and Sungjin Ahn. Extendable planning via multiscale diffusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026a.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1c92edb990a05f2269f0cc3afbb4c952-Abstract-Conference.html.

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*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.045.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. DataMIL: Selecting data for robot imitation learning with datamodels. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/033d9e8dbbbc0ad90e59222cf1db0fc2-Abstract-Conference.html.

Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.011.

Scott Emmons, Ajay Jain, Michael Laskin, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5251–5262, 2020.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620, 2022.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag R. Sanketi, Dhruv Shah, Coline Manon Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.023.

Charles A. Hepburn and Giovanni Montana. Model-based trajectory stitching for improved behavioural cloning and its applications. *Machine Learning*, 113:647–674, 2024. doi: 10.1007/s10994-023-06392-z.

Jonathan Ho, Ajay N. Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Zixuan Huang, Yating Lin, Fan Yang, and Dmitry Berenson. Subgoal diffuser: Coarse-to-fine subgoal generation to guide model predictive control for robot manipulation. *arXiv preprint arXiv:2403.13085*, 2024.

Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled sailor: Robust imitation via learning to search. *CoRR*, abs/2506.05294, 2025.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. DexMimicGen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025. doi: 10.1109/ICRA55743.2025.11127809.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993. URL https://people.csail.mit.edu/lpk/publications.html.

Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. CCIL: Continuity-based data augmentation for corrective imitation learning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LQ6LQ8f4y8.

Sungyoon Kim, Yunseon Choi, Daiki E. Matsunaga, and Kee-Eung Kim. Stitching sub-trajectories with conditional diffusion model for goal-conditioned offline rl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13160–13167, 2024. doi: 10.1609/aaai.v38i12.29215.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AP1MKT37rJ.

Kyowoon Lee and Jaesik Choi. State-covering trajectory stitching for diffusion planners. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. DiffStitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28597–28609. PMLR, 2024.

Yunhao Luo, Utkarsh A. Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. In *Advances in Neural Information Processing Systems*, volume 38, 2025. Spotlight.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690. PMLR, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In *International Conference on Learning Representations*, 2025.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned rl with latent states as actions. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Austin Patel, Ben Pekarek, Joel Enrique Castro Hernandez, and Shuran Song. Behavior prompting policy: Demonstrations as prompts for manipulation. *arXiv preprint arXiv:2606.30457*, 2026. URL https://arxiv.org/abs/2606.30457.

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018. URL https://pathak22.github.io/zeroshot-imitation/resources/iclr18.pdf.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Juntao Ren, Gokul Swamy, Steven Wu, Drew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42428–42448. PMLR, 2024. URL https://proceedings.mlr.press/v235/ren24c.html.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Chaining behaviors from data with model-free reinforcement learning. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 2162–2177. PMLR, 2021. URL https://proceedings.mlr.press/v155/singh21a.html.

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Steven Wu, and Drew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e77d684aae157abd84df1eeb76d8b9cd-Abstract-Conference.html.

Xin Tong, Yang Feng, and Jingyi Jessica Li. Neyman–pearson classification algorithms and NP receiver operating characteristics. *Science Advances*, 4(2):eaao1659, 2018. doi: 10.1126/sciadv.aao1659. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC5804623/.

UMI Robot Dataset Community. UMI Robot Dataset Community. https://umi-data.github.io/, 2026. Accessed July 2026.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pp. 3024–3030, 2015. doi: 10.1609/aaai.v29i1.9590. URL https://ojs.aaai.org/index.php/AAAI/article/view/9590.

Shuze Wang, Yunpeng Mei, Hongjie Cao, Yetian Yuan, Gang Wang, Jian Sun, and Jie Chen. Robust offline imitation learning through state-level trajectory stitching. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13497–13504. IEEE, 2025.

Kun Wu, Ning Liu, Zhen Zhao, Di Qiu, Jinming Li, Zhengping Che, Zhiyuan Xu, and Jian Tang. Learning from imperfect demonstrations with self-supervision for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16899–16906, 2025a. doi: 10.1109/ICRA55743.2025.11127918.

Yilin Wu, Thomas Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: VLM-in-the-loop policy steering via latent alignment. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.076. URL https://www.roboticsproceedings.org/rss21/p076.html.

Zhao-Heng Yin and Pieter Abbeel. Offline imitation learning through graph search and retrieval. In *Proceedings of Robotics: Science and Systems*, 2024.

Hang Yu, Di Zhang, Qiwei Du, Yanping Zhao, Hai Zhang, Guang Chen, Eduardo E. Veas, and Junqiao Zhao. ASTRO: Adaptive stitching via dynamics-guided trajectory rollouts. *arXiv preprint arXiv:2511.23442*, 2025.

Sheng Yue, Jiani Liu, Xingyuan Hua, Ju Ren, Sen Lin, Junshan Zhang, and Yaoxue Zhang. How to leverage diverse demonstrations in offline imitation learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58037–58067, 2024.

Xiaoyu Zhang, Matthew Chang, Pranav Kumar, and Saurabh Gupta. Diffusion meets DAgger: Supercharging eye-in-hand imitation learning. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.048.

Zhaoyi Zhou, Chuning Zhu, Runlong Zhou, Qiwen Cui, Abhishek Gupta, and Simon Shaolei Du. Free from bellman completeness: Trajectory stitching via model-based return-conditioned supervised learning. In *International Conference on Learning Representations*, 2024.

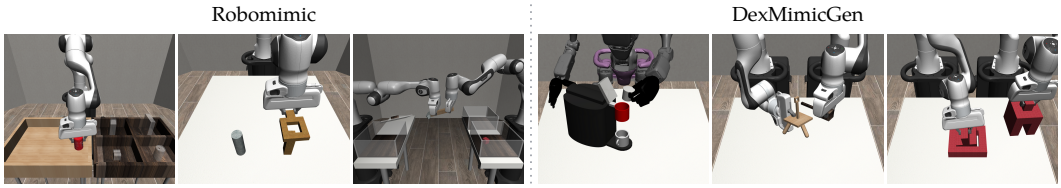

Figure 11: **Simulation benchmarks.** We evaluate dataset rewriting on three Robomimic tasks with suboptimal demonstrations (Mandlekar et al., 2022) and three DexMimicGen tasks with near-optimal demonstrations (Jiang et al., 2025).

A Datasets and Task Setups

Below we provide more detail about our real-world tasks, evaluation procedure, and dataset composition in both real and sim.

A.1 Real-Robot Tasks

All demonstrations are collected with handheld UMI devices, specifically iPhUMI (Patel et al., 2026), and policies are given only RGB wrist-camera views with relative proprioception from the starting position of each trajectory. There is no shared-coordinate reference frame, and thus no absolute proprioception that can be used/shared across demonstrations.

Sweater folding. In this task, the robot first folds each sleeve toward the center of the sweater, then folds the bottom upward. This task tests whether the policy can complete a sequence of coordinated manipulations of a deformable object, and is a longer-horizon task where mistakes may compound. We collect a total of 108 successful trajectories, and 78 failure trajectories for this task.

Dish racking. In this task, the robot picks up a dish with its left hand, moves towards the center for a handover and reorientation to the right hand, then moves to place the plate into the rack. The handoff requires the receiving grasp to support the dish before the other gripper releases it, and also reorients the plate in the gripper for insertion into the rack to be physically feasible. This task requires high precision: the initial pickup of the thin rim around the plate, the handover coordination, and the insertion between poles not much wider than the plate itself. We collect a total of 179 successful trajectories and 59 failure trajectories for this task.

Evaluation. We deploy all policies on a bimanual ARX-5 arm setup. For fairness, we fix 50 starting states per task, which we give to all methods. We use image overlays with the two arms at the same starting pose to reset each object to a precise pose.

A.2 Simulation Tasks

Robomimic. For each Multi-Human task, we subsample 40 successful demonstrations, with 50% from the most suboptimal pool and 50% from the most optimal pool. The minimum/median/maximum trajectory lengths are 98/227/1,050 steps on *Can*, 123/311/1,051 on *Square*, and 392/676.5/2,614 on *Transport*. All methods receive the same selected demonstrations; methods that use failure data also receive the same fixed set of 150 unsuccessful policy rollouts per task. All unsuccessful policy rollouts were generated by three randomly selected Diffusion Policy checkpoints, trained with different configuration than the baselines reported in all experiments.

DexMimicGen. We evaluate *Coffee*, *Three Piece Assembly*, and *Threading* using 100 successful demonstrations per task, with the same additional 200 unsuccessful episodes available to methods that use failure data. These demonstrations provide a comparison with the suboptimal Robomimic mixture: the recorded successful behavior is already near-optimal, leaving less opportunity to shorten its continuation.

Following prior evaluation practice (Chi et al., 2023), we evaluate checkpoints every 25 epochs on the same 50 fixed evaluation initial conditions and report the mean and standard error of the maximum success rate observed across seeds. All methods use the same checkpoint-evaluation frequency

and the same fixed evaluation starts. We tune the baselines extensively rather than relying on their released configurations; Figure 16 shows that our tuning substantially improves their performance.

B Additional Results and Ablations

B.1 Does Rewriting Reduce Retraced Motion on the Real Robot?

Needle retraces less gripper travel across the two tasks. We measure the fraction of gripper travel that re-enters a 2 cm region previously visited and subsequently left during the same rollout, using the same 50 evaluation trials per task and policy as the main results. Pooled over travel across both tasks, this fraction is 9.2% for Needle, compared with 10.8% for DP and 16.7% for AWR (Figure 12). Travel pooling gives longer paths more weight; averaging the fraction over trials provides a complementary summary.

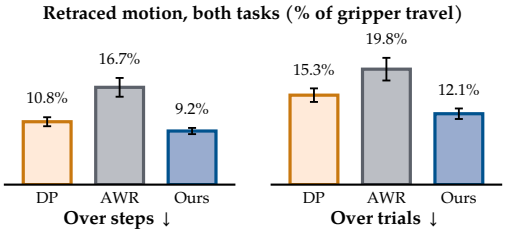

Figure 12: **Retraced motion on the real robot.** Fraction of gripper travel revisiting a previously exited 2 cm region, over 100 trials per policy (50 per task). Left: pooled over steps. Right: averaged over trials. Bars show the estimate and standard error.

B.2 Can Stitching Improve With a Subset of Stages?

The full design draws on three kinds of connections: skipping ahead within a successful demonstration (S1), joining a different successful demonstration (S2), and connecting a failed attempt to a successful demonstration (S3). We seek to better understand the contributions from training on subsets of this combination. Figure 13 shows how well a policy learns with bridges when trained on each stage individually, and from S1+S2 without failure-to-success bridges. We select these subsets from the same bridge pool and retain the original successful demonstrations.

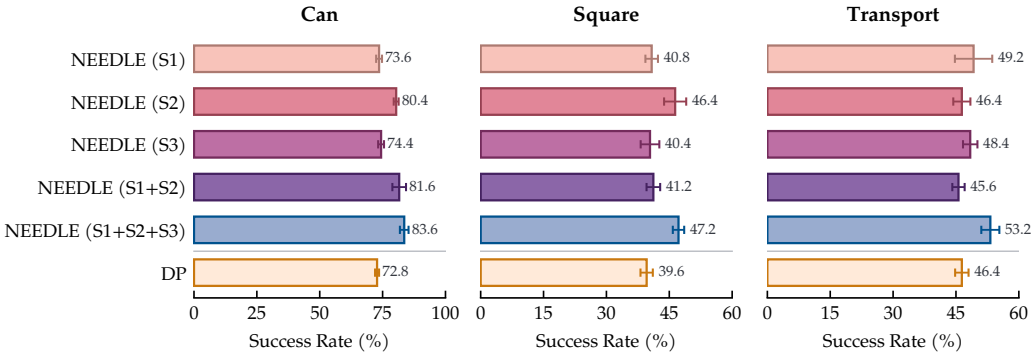

Figure 13: **Which connections help the policy?** Stage subsets compared with the main-paper DP and Needle references. Bars show mean success and standard error over five training seeds.

No single stage reaches the three-stage configuration on all three tasks. Cross-demonstration bridges alone (S2) reach 80.4% on *Can* and 46.4% on *Square*. Adding failure-to-success bridges to S1+S2 raises success from 41.2% to 47.2% on *Square* and from 45.6% to 53.2% on *Transport*. Thus, the relative value of the stages differs across tasks; these results do not imply that each stage is necessary everywhere.

B.3 What is the best way to train on stitched data?

A verified bridge provides a new action continuation from a recorded observation, but it does not create a complete synthetic trajectory. In particular, its endpoint may differ slightly from the recorded target, or be from a different coordinate system, and taking the shortcut does not eliminate the possibility that the deployed policy will visit states along the original trajectory. Our training construction therefore (i) retains the original actions alongside the bridge, (ii) preserves supervision from bypassed segments, and (iii) stops the new action target at the bridge endpoint rather than concatenating the target trajectory as if the connection were exact. Figure 14 ablates these choices using the same IDM and verifier as trained in the main paper. We also provide detailed visualizations of the key differences between these sampling methods on our website.

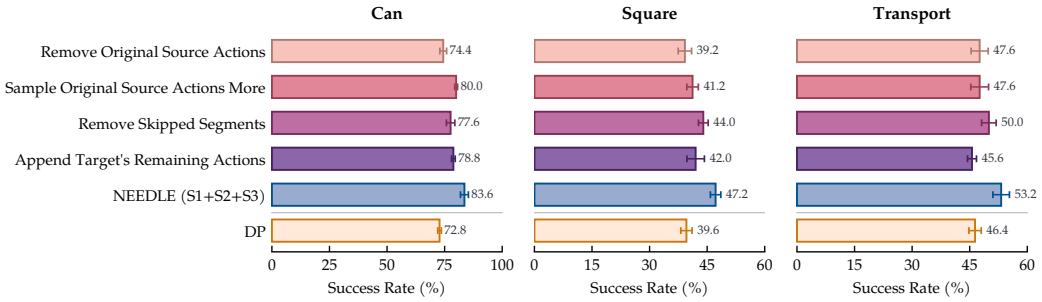

Figure 14: **How should a policy learn from a bridge?** Rows below NEEDLE name alternative ways to construct training examples. Bars show mean success and standard error over five training seeds; DP and NEEDLE are main-paper references.

Keep both ways of acting from the bridge's starting point. Our design retains them as another example of how to act from the same observation. The first alternative removes this example, leaving only the bridge. The second keeps it but doubles its relative sampling weight (0.5 to 1.0), making the recorded actions more common relative to bridge actions. These rows distinguish keeping useful original behavior from simply showing it more often.

Keep learning from skipped segments. A shortcut does not teach the policy what to do at the observations along the skipped segment. We retain those original observation–action examples so the policy can still learn from them. The alternative removes them. It reaches 50.0% on *Transport*, compared with the 53.2% reference, with standard errors of 1.9 and 2.2 points respectively.

Stop the training target at the bridge endpoint. We mask action loss when a training window includes both bridge actions and the target demonstration's actions, and learn the target demonstration's remaining actions separately, paired with its own observations. The alternative appends those actions immediately after the bridge, treating the connection as exact. This is possible in simulation's shared coordinate frame, but may be unreliable when a bridge lands slightly away from its target. Even so, the alternative reaches 42.0% on *Square* and 45.6% on *Transport*, below their reference values of 47.2% and 53.2%.

B.4 Do the bridge actions themselves matter?

Removing every training window that contains bridge actions, while keeping the reweighted original windows, lowers success (Figure 15) from 83.6% to 77.2% on *Can*, from 47.2% to 42.8% on *Square*, and from 53.2% to 45.6% on *Transport*, so a large part of the gain over DP comes from the bridge actions rather than from reweighting the recorded data.

B.5 Full DexMimicGen Results

Table 2 expands the near-optimal-demonstration comparison to all evaluated baselines on the three DexMimicGen tasks. All methods draw from a fixed corpus of 100 successful demonstrations; methods that use unsuccessful data receive the same additional 200 failure episodes.

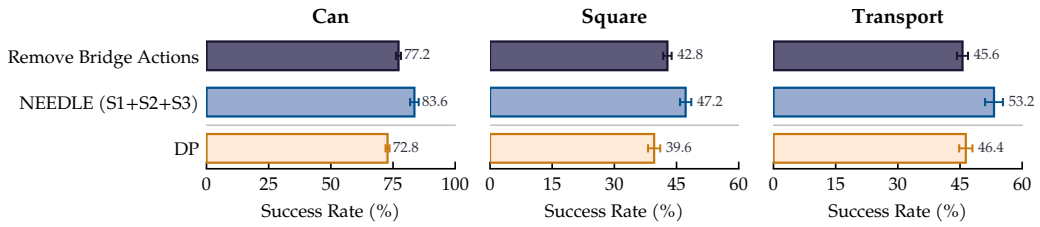

Figure 15: **Do the bridge actions matter?** The first row removes every training window that contains bridge actions and keeps the reweighted original windows. Bars show mean success and standard error over five training seeds; DP and NEEDLE are main-paper references.

NEEDLE has the highest mean success on *Coffee* and *Three Piece Assembly*, while AWR performs best on *Threading*. Together with the DP comparison in the main text, these results show that rewriting can still provide modest gains when demonstrations are near-optimal, although the benefit is smaller and task-dependent.

Task	DP	SBR	AWR	IDQL	SARM	**Ours**
Coffee	62.0 $\pm$1.2	60.7 $\pm$2.4	52.7 $\pm$1.8	49.3 $\pm$1.8	39.3 $\pm$1.8	**63.3** $\pm$0.7
Three Piece Assembly	51.3 $\pm$0.7	44.7 $\pm$1.8	47.3 $\pm$1.3	43.3 $\pm$3.5	26.7 $\pm$1.8	**55.3** $\pm$1.8
Threading	46.0 $\pm$1.2	40.7 $\pm$2.4	**50.0** $\pm$1.2	40.0 $\pm$1.2	34.0 $\pm$2.0	46.0 $\pm$1.2

Table 2: **Full baseline comparison on DexMimicGen.** Success rates (%) reported as mean and standard error over three training seeds with 50 evaluation initial conditions per seed. NEEDLE has the highest mean success on two tasks; AWR leads on Threading.

C BASELINE ADAPTATIONS AND TUNING

C.1 SHARED SETUP

We adapt each baseline to the diffusion-policy backbone used by NEEDLE and DP, while retaining its data-selection, weighting, or action-reranking mechanism. The adapted policies share cached DINOv3 features (Siméoni et al., 2025), proprioception, and prediction and execution horizons. Figure 16 shows that these adaptations substantially improve success over the released-policy implementations evaluated on the same observations. The comparison includes both changes to the policy architecture and task-specific tuning; it assesses the baselines in our visual-control setting.

C.2 BASELINE-SPECIFIC ADAPTATIONS

SBR. SBR retrieves transitions from unsuccessful demonstrations using their similarity to successful demonstrations and weights the retrieved data for policy training. We vary the retrieval representation (raw DINOv3 features or learned autoencoder latents), weighting rule (flat or one-stream), inclusion of proprioception in matching, and use of the autoencoder's internal validation split. All selected configurations remove this internal split, which otherwise excludes four of the 40 successful demonstrations from policy training. Raw-feature retrieval returns no failure transitions on Can and Transport; the selected configurations use autoencoder latents on these tasks.

AWR. AWR (Peng et al., 2019) weights policy-training windows by $\min(\exp(\beta(Q - V)), 100)$, using advantages from an IQL critic (Kostrikov et al., 2022). We apply this weighting rule to the shared diffusion actor in place of the released Gaussian MLP actor. The critic is trained once and frozen, and the same critic checkpoints are used by IDQL. We compare IQL's antmaze temperature $\beta = 10$ (IDQL's default is 3) with temperatures calibrated to target effective sample-size (ESS) fractions of 0.5 or 0.8, computed over per-timestep advantages of either the full dataset or the successful demonstrations. We also vary the actor and critic training corpora independently between successful demonstrations alone and the combined success and failure dataset. The selected actors

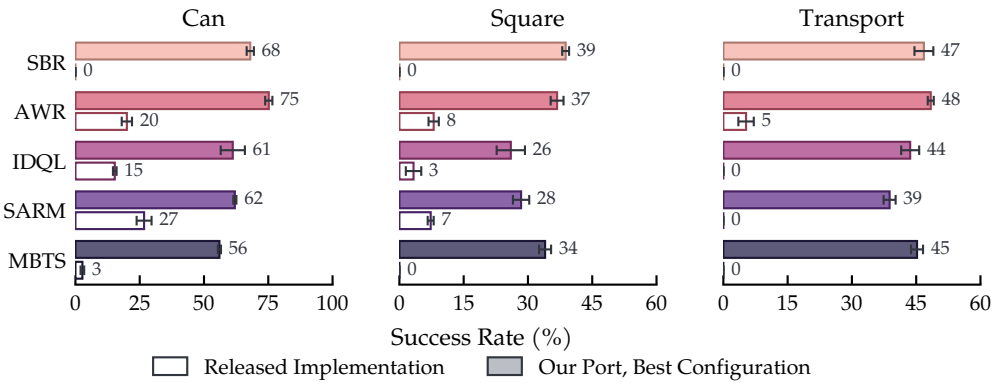

Figure 16: **Adapting baselines to visual robot control.** Hollow bars show released-policy implementations evaluated on cached DINOv3 features and proprioception; filled bars show the adapted and tuned configurations used in Figure 6. Adaptation improves success across tasks. Error bars show standard errors over three training seeds for original baseline implementations, and five training seeds for our adaptation; zero-height bars indicate measured zero success.

use successful demonstrations only; the selected critics use all episodes on Can and Transport and successful demonstrations only on Square.

IDQL. IDQL uses a critic to rerank action samples from a diffusion policy. We vary the number of candidates (1, 16, or 64), selection rule (maximum Q-value or expectile resampling), critic-checkpoint selection rule, and actor and critic training corpora. The selected configuration on all three tasks uses 16 candidates, maximum-Q selection, a success-only actor, and the critic checkpoint with the highest held-out ranking accuracy. We additionally evaluate the same actor without reranking to assess the contribution of critic-based selection.

SARM. SARM weights policy-training windows using learned visual progress. Its original formulation uses annotated subtask boundaries and language embeddings, which are unavailable in our experimental setting. We therefore evaluate SARM, an adaptation without language conditioning, using either whole-episode progress prediction or a stage-aware model with proportional temporal boundaries. We compare these models with the released rewind augmentation. The selected configurations use the stage-aware model on Can and rewind augmentation on Square and Transport. Proportional boundaries provide temporal structure rather than semantic subtask annotations, so these results characterize the adapted method under our supervision constraints.

MBTS. MBTS rewrites the offline dataset before policy training: for each transition it retrieves nearby states from other trajectories, proposes a joining action with a learned one-step inverse model, and accepts the splice only if learned dynamics, a density check, and a value-improvement gate all pass, repeating this for several passes. We replace its privileged simulator state (unavailable in our setup) with the same frozen DINOv3 features and proprioception the other baselines use, and reward is the sparse episode-success label rather than a simulator reward. We vary the actor class (the released deterministic MLP versus our diffusion backbone) and train the diffusion actor on the five-pass rewritten corpus. The reported configuration on all three tasks is that rewritten corpus.

D Additional Implementation Details

D.1 Shared Architecture and Optimization

Visual representation. Across Robomimic and real-robot experiments, we use a frozen DINOv3 ViT-B/16 encoder (Siméoni et al., 2025) for the downstream diffusion policy, baselines, IDM, and verifier. The optimizer settings and observation and action horizons in Table 3 are also shared across these domains. The shared encoder provides visual features for both learning the recorded behavior and assessing candidate bridges, and allows us to cache image features during training. The IDM

and downstream policy pool the image patches into spatial features, while the verifier retains a grid of tokens so that it can attend to different parts of the source and target images.

Setting	Value
Policy, IDM, and verifier	
Image encoder / resolution	Frozen DINOv3 ViT-B/16 / 224×224
Optimizer / learning rate	AdamW / 10^{-4}
Weight decay	10^{-6}
Adam coefficients / epsilon	$(0.95, 0.999)$ / 10^{-8}
Learning-rate schedule / warmup	Cosine decay / 500 steps
Diffusion policy and IDM	
Observation history / prediction horizon	2 / 24 steps
Training noise steps	50
Downstream diffusion policy	
Action execution horizon	12 steps
Batch size	64
Training epochs (Robomimic / real robot)	300 / 500

Table 3: **Shared visual and training settings.** The encoder and optimizer settings apply to the policy, IDM, and verifier; the remaining rows specify their scope.

D.2 Inverse Dynamics Model

Prediction and execution horizons. The 24-row action prediction and the 23-step bridge horizon refer to different parts of the same sequence. With two observation steps, one predicted row aligns with the earlier observation, leaving 23 actions from the current source onward. The downstream policy executes 12 actions before replanning. For IDM training, the remaining rows after reaching a sampled goal hold the target pose; they do not continue along the demonstration past that goal. Table 4 collects the architecture and training settings.

Setting	Value
Architecture	
Visual feature pooling	Spatial softmax over a 14×14 patch grid
DDIM sampling steps (model configuration)	50
Maximum executable bridge length	23 actions
Action representation	Absolute pose (simulation); source-relative pose (real robot); 6D rotation and gripper command
U-Net channel widths	512, 1024, 2048
Diffusion-step embedding dimension	256
U-Net kernel size / normalization groups	5 / 8
Goal-conditioning MLP hidden widths	256, 256
Optimization	
Batch size	128

Table 4: **IDM architecture and hyperparameters.** Shared visual and optimizer settings are in Table 3.

D.3 Verifier

The verifier learns from positive source–goal pairs connected by recorded actions and negative pairs that those actions do not connect. Negative examples include goals beyond the action horizon and in other trajectories. Sampling determines which examples the verifier sees, while the loss assigns equal total weight to the positive and negative terms. Increasing the number of negatives per positive therefore does not increase their aggregate loss weight. Table 5 collects the architecture, sampling, loss, and calibration settings.

Setting	Value
Architecture	
Source and target visual tokens	7×7 patch grid per camera
Transformer layers / attention heads / hidden width	2 / 4 / 512
Feed-forward width / dropout	4096 / 0
Attention / positional encoding	Causal / rotary
Proprioception / action normalization	Source only / per action dimension
Maximum scored action prefix	23 actions
Training pairs and loss	
Positive anchors / negatives per group	1 / 7
Spacing between candidate targets in one episode	23 steps
Loss function	Balanced binary cross-entropy
Positive / negative aggregate loss weight	0.5 / 0.5
Optimization and calibration	
Batch size / training epochs	1024 / 40
Initial calibration target false-positive rate	0.01

Table 5: **Verifier architecture and hyperparameters.** Sampling and training values describe the Robomimic runs. Shared visual and optimizer settings are in Table 3.

D.4 Stitching and Training-Window Construction

Why keep one bridge per source? A source state can have several candidate targets. Keeping all of them would make that state's contribution depend on how many matches happened to be found and could give the policy several competing action targets at the same departure. We instead retain the shortest bridge-plus-suffix path for each source. For successful sources, this also maximizes the planned reduction in remaining steps, which must be positive. This separates choosing a useful continuation from deciding how often to train on it.

Setting	Value
Candidate search and acceptance	
IDM proposals per pair	8
Bridges retained per source	1
Successful-source preference	Shortest bridge-plus-suffix path; positive steps saved
Failure-source preference	Shortest bridge-plus-suffix path
Relative sampling weights per window	
Other recorded windows	0.5
Recorded windows skipped by a within-trajectory bridge	0.5
Recorded action window at a bridge source	0.5
Recorded suffix after the target (S1 / S2)	1.0 / 1.0
Recorded failure-prefix windows	0

Table 6: **Stitch generation and training-window construction.** Shared settings for the nominal Robomimic runs; task-specific candidate budgets appear in Table 7. Weights control how often a window is sampled, rather than scaling its action loss.

Table 7 lists the task-specific candidate budgets used for the main Robomimic comparison.

D.5 Compute Resources

All runs use a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU unless noted; earlier development runs also used NVIDIA L40S GPUs. In simulation, the IDM trains for 300 epochs on *Can* and 150 epochs on *Square* and *Transport* in 2–4 h, the verifier for 40 epochs in 6–11 GPU-hours, offline stitching of the full candidate pool takes about 2 h, and the downstream policy trains for 300 epochs with periodic simulator evaluation in 2–9 h depending on the task. The IDM and verifier can be trained in parallel on separate GPUs. Baseline policies cost the same as our downstream policy since they share the training recipe.

Setting	Can	Square	Transport
Observations, data, and evaluation			
RGB cameras	External + wrist	External + wrist	2 shoulder + 2 wrist
Action dimensions	10	10	20
Successful / unsuccessful episodes	40 / 150	40 / 150	40 / 150
Full evaluation budget	500	500	700
Evaluation starts per seed	50	50	50
Task-specific stitching choices			
IDM training epochs	300	150	150
Candidate cap, each of S1 and S2	10^5	10^5	10^5
Failure-candidate cap	10^6	10^5	10^5
Failure targets per source	16	8	8

Table 7: **Robomimic task settings.** Columns describe the nominal policies used for the main success comparison. Shared training settings appear in Table 3.

D.6 Evaluation and Diagnostic Protocols

Policy success and completion budgets. Robomimic policies are trained for 300 epochs and evaluated every 25 epochs on 50 initial conditions, paired across baselines. Figure 6 reports success and standard error over five training seeds evaluated for 50 episodes each; completion steps are measured at that same evaluation point and averaged over successful rollouts. For the rollout panels in Figure 8, we evaluate the selected policies from 50 matched held-out initial conditions per task and training seed. A trial contributes to success at budget b only if its first success occurs by step b. Full evaluation budgets are 500 steps for *Can* and *Square* and 700 for *Transport*, which we take directly from the Multi-Human setting in the Robomimic benchmark (Mandlekar et al., 2022).

Bridge continuations. For the continuation panels in Figure 8, we restore the same simulator state for each source state that ends up with a proposed stitch, execute either the recorded actions or the selected bridge for the bridge's duration, and resume the same DP policy from each landing. We measure remaining steps from the end of this action sequence. The comparison uses the fastest successful resume; the fitted distributions therefore describe completion from jointly successful resumes.

Recovery from failure states. The rollout panels in Figure 9 use 50 sampled failure states per task, shared across methods and evaluated with five training seeds and 16 resumes per state. We first compute the success fraction across the 16 resumes from each state, then average over states and training seeds. We focus on failure states in which the object remains in the robot's reachable workspace, excluding cases such as Transport objects dropped to the ground. Each resume receives a full task horizon.