REDFLOW: REDIRECT FAILURE INTO ACTION-LEVEL CORRECTIONS FOR FLOW-MATCHING VLA POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) can improve Vision-Language-Action (VLA) policies through deployment experience, but reward- and preference-based RL primarily identifies desirable behaviors without specifying how to correct failed actions, leaving failure trajectories underutilized and limiting sample efficiency. Can such corrections be derived from fixed rollouts? Our key insight is that rollouts with different outcomes may contain action chunks executed in similar states, allowing higher-quality chunks to serve as locally supported corrective targets. Building on this insight, we introduce **RedFlow**, an offline post-training method for flow-matching VLA policies. *Execution-Context Matching* groups chunks using a compact representation of estimated task progress and the robot's proprioceptive state. *Quality-Guided Action Redirection* assigns signed quality scores to chunks and aggregates higher-quality chunks into corrective targets, reinforcing high-quality chunks, suppressing low-quality chunks, and redirecting correctable chunks toward their targets. RedFlow requires neither human intervention nor online data collection during post-training. Across four LIBERO suites, RedFlow improves the average success rate from 56.2% to 68.2%, outperforming the strongest evaluated offline baseline, AWR (62.3%), by 5.9 percentage points. Across three real-robot tasks, it improves the average success rate from 56.7% to 74.7%. On LIBERO-Spatial, RedFlow achieves a success rate of 75.8% with 1,536 fixed rollouts, while the evaluated online methods require 8.7–16 times as many fresh post-training rollouts to reach the same threshold.

1 INTRODUCTION

Reinforcement learning (RL) (Levine, 2018) can, in principle, improve Vision-Language-Action (VLA) policies beyond supervised imitation by learning from deployment experience. Recent work has explored reward- and preference-based policy optimization to improve VLA behavior (Lu et al., 2025; Huang et al., 2025; Zhang et al., 2026; Chen et al., 2026b; Zhang et al., 2024; Li et al., 2025; Wang et al., 2026a). However, these approaches remain *evaluative*: they assess which actions or trajectories are better or worse, but do not provide *corrective* supervision that specifies how actions in failure trajectories should be modified to achieve success. Consequently, the policy may learn to avoid undesirable behaviors without learning what to do instead, leaving failure trajectories underutilized and limiting sample efficiency.

A direct way to provide corrections is human-in-the-loop (HIL) learning, where a human intervenes when the flow-matching VLA policy begins to deviate during execution and provides corrective demonstrations (Intelligence et al., 2025; Yu et al., 2026b; Wu et al., 2026; Li et al., 2026). Such intervention provides explicit corrective actions that evaluative feedback does not specify, reducing the need for additional policy rollouts to discover better actions. While effective, HIL requires repeated expert intervention, making it difficult to scale across tasks, rollouts, and robotic platforms. This motivates a natural question: can we derive corrections from limited rollout experience itself, without relying on external human intervention?

To answer this question, our key insight is that if a failed trajectory shares an identical state (including the same robot pose, object positions, and robot–object contact configuration at the same stage of the task) with a successful trajectory, it can be corrected by stitching the successful trajectory's action sequence from that state onward onto the failed trajectory.

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.

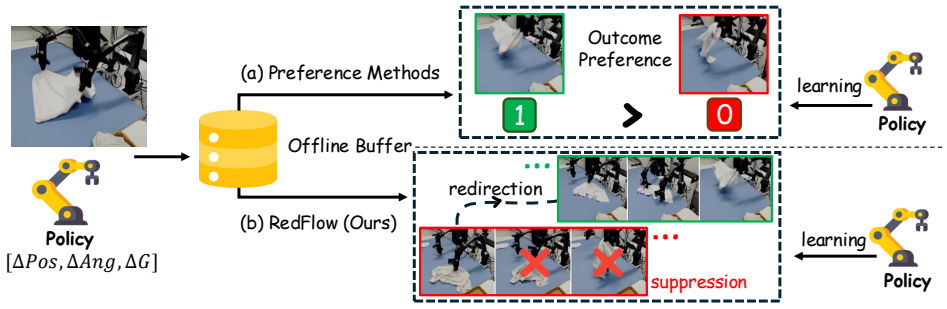


Figure 1: **RedFlow learns from mixed-quality rollouts through action-level guidance.** (a) Outcome-level preferences rank successful and failed rollouts but do not, by themselves, identify which action chunks to change or provide alternatives. (b) RedFlow suppresses low-quality chunks and redirects selected predictions toward locally supported references aggregated from higher-quality actions in comparable execution contexts.

Turning this insight into a practical learning framework raises two key challenges. First, identical states are rarely encountered across trajectories in practice. For example, across two rollouts to grasp a cup, the gripper may approach from different angles or the cup may shift slightly. As a result, most states in failed trajectories may not exactly match any state in successful trajectories, preventing failure trajectories from being corrected and thereby limiting their exploitation. Second, stitching a failed trajectory to a successful one at an identical state provides a trajectory-level correction by replacing the entire remaining action sequence. Yet an identical state identifies a valid stitching point, not necessarily where the failed trajectory went wrong. Such replacement does not reveal which specific actions lead to failure or how they should be modified to achieve success.

To address these challenges, we propose **RedFlow**, an offline post-training framework that constructs action-level corrective supervision from mixed-quality rollout experience for flow-matching VLA policies. First, although identical states are rare across trajectories, successful and failed rollouts may still encounter comparable execution contexts. We therefore propose *Execution-Context Matching*, which combines estimated task progress with robot proprioception to identify action chunks at similar task stages and robot configurations. This compact representation provides local cross-trajectory support without assuming that matched chunks have identical visual, object, or contact states. Second, to move beyond trajectory-level stitching, we introduce *Quality-Guided Action Redirection*, which assigns each action chunk a retrospective quality score and redirects selected low-quality predictions rather than replacing the entire remaining action sequence. Higher-quality actions from matched execution contexts are aggregated into a locally supported corrective reference, while the policy is also discouraged from repeating low-quality actions. The resulting supervision provides locally supported guidance for action-space redirection when comparable execution contexts recur. Figure 1 contrasts RedFlow with outcome-level preference learning: rather than only ranking rollout outcomes, RedFlow suppresses low-quality action chunks and redirects selected predictions toward locally supported corrective references.

Our contributions are summarized as follows: **(i)** We enable flow-matching VLA policies to turn a fixed mixed-quality rollout buffer into action-level corrective references. RedFlow identifies low-quality action chunks, finds comparable execution contexts in which higher-quality actions were observed, and uses those actions to provide locally supported guidance for policy redirection, without additional human corrections or online data collection during post-training. **(ii)** We demonstrate consistent gains in simulation and on real robots. Across the four LIBERO suites, RedFlow improves the average success rate from **56.2%** to **68.2%** (+12.0 points) and exceeds the strongest evaluated offline baseline, AWR (62.3%), by **5.9 points**. Across three real-robot tasks, it improves the average from **56.7%** to **74.7%** (+18.0 points), including a 31.0-point gain on clothes folding. On LIBERO-Spatial, RedFlow reaches 75.8% success with a fixed buffer of 1,536 rollouts, whereas the evaluated online methods require 13,312–24,576 fresh rollouts to reach the same threshold, corresponding to **8.7–16× as many additional post-training rollouts**.

2 Related Work

RL for VLA Policy RL methods use rewards, preferences, or value estimates to judge which behavior is better and improve the policy accordingly. π_{RL} and π-StepNFT use online rollout feedback to improve VLA policies (Chen et al., 2026a; Wang et al., 2026a); ReinFlow studies online RL for flow-matching robot policies (Zhang et al., 2025). AWR, CQL, and IQL provide general offline RL methods (Peng et al., 2019; Kumar et al., 2020; Kostrikov et al., 2022), while Co-RFT and ARFM adapt offline RL to VLA policies (Huang et al., 2025; Zhang et al., 2026). DICE-RL uses online feedback and value-guided selection to improve individual actions (Sun & Song, 2026). Although these methods can improve individual actions, their signals are primarily *evaluative*: they favor some behavior over others but do not themselves specify an observed alternative for a failed chunk. RedFlow instead matches execution contexts in a fixed mixed-quality buffer to obtain locally supported *corrective* references for selected negative chunks.

Human-Provided Corrections. One way to tell a policy what to do instead is to ask a human to intervene when the robot deviates (Li et al., 2026). FlowPRO uses paired intervention-and-rollback trajectories to post-train a flow-matching VLA policy offline (Wu et al., 2026). These methods provide explicit corrective actions, but require human intervention data. RedFlow instead derives corrective references from the policy's existing mixed-quality rollouts.

Action Guidance from Collected Experience. Action guidance can also come from learned models or previously collected experience. Q-VGM constructs flow-velocity targets from a learned action-value critic (Wang et al., 2026b). ForesightFlow ranks generated candidate actions using a learned success potential at inference time (Mei et al., 2026), while SAILOR searches for recoveries at test time using learned world and reward models (Jain et al., 2025). Trajectory stitching connects offline trajectories through generated actions or transitions (Hepburn & Montana, 2022; Li et al., 2024). Thus, guidance can be obtained from critic gradients, inference-time candidate selection, or generated trajectory connections. RedFlow instead matches action chunks in a fixed buffer by task progress and proprioception, then uses higher-quality observed actions in the same execution context to form a soft reference for selected negative chunks. This reference guides training, alongside quality-weighted imitation and failure suppression. It is not a verified action that could simply be executed from the negative state, nor a trajectory splice. The frozen progress model estimates quality and helps match chunks; it does not generate corrective actions (Xu et al., 2026).

3 Methodology

3.1 Preliminaries

Problem Setup. We formulate language-conditioned robotic manipulation as an MDP over an observation space $\mathcal{O}$ and an action space $\mathcal{A}$. At each action-chunk step t, given a task instruction $l \in \mathcal{L}$, the agent receives an observation $o_t = (I_t, q_t)$, where $I_t \in \mathcal{I}$ denotes the visual input and $q_t \in \mathbb{R}^{D_q}$ denotes the proprioceptive state. A flow-matching VLA policy $\pi_\theta(a_t \mid o_t, l)$ predicts an action chunk $a_t \in \mathbb{R}^{K \times D}$, which consists of K consecutive D-DoF commands. The robot executes this chunk before the agent receives the next action-chunk observation o_{t+1}. An episode yields a trajectory $\tau = \{(o_t, a_t)\}_{t=0}^{T-1}$ and a binary outcome label $y_\tau \in \{0, 1\}$. We refer to τ as a *successful trajectory* when $y_\tau = 1$ and as a *failed trajectory* otherwise.

Optimization Objective. The goal of offline post-training is to improve the deployment policy with respect to expected task return. For a language-conditioned manipulation task, the standard RL objective is

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t r(o_t, a_t, l)\right], \tag{1}$$

where $r(o_t, a_t, l)$ denotes the task reward and may be sparse or dominated by the terminal success outcome. At the action-chunk level, local policy improvement is governed by the standard action advantage

$$A^\pi(o_t, a_t, l) = Q^\pi(o_t, a_t, l) - V^\pi(o_t, l). \tag{2}$$

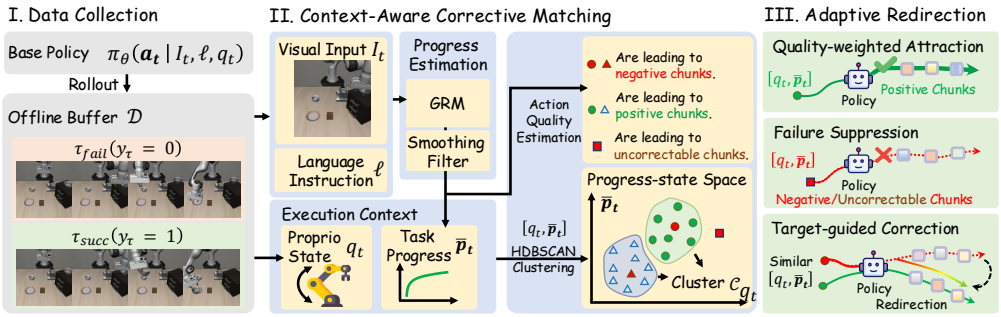


Figure 2: **RedFlow pipeline.** RedFlow consists of three stages: **(I) Data Collection**, where successful and failed rollouts are stored in an offline buffer; **(II) Execution-Context Matching**, which uses estimated task progress and proprioceptive states to define execution contexts and groups similar contexts with HDBSCAN; and **(III) Quality-Guided Action Redirection**, which estimates chunk quality, constructs corrective targets within matched contexts, reinforces high-quality chunks, suppresses low-quality chunks, and redirects correctable negative chunks toward their targets.

Positive-advantage actions are preferred under policy improvement, whereas negative-advantage actions should receive less probability mass. However, in our offline setting, the quantity A^π is unavailable because the buffer provides only coarse trajectory-level outcomes rather than direct action-level credit assignment. We therefore construct a signed chunk-quality score using learned task progress and trajectory outcomes. This score is not a policy advantage and is not intended to estimate the causal effect of an individual action chunk.

Flow-Matching VLA Policy. The idea of constructing corrective supervision from rollout experience is not specific to flow matching. In this work, we focus on flow-matching VLA policies because they are widely used in recent robotic manipulation systems.

We instantiate π_θ as a conditional flow-matching model (Lipman et al., 2023). Given an observation o_t and a task instruction l, the policy generates an action chunk $a_t = x_0$ by transporting a noise sample $x_1 \sim \mathcal{N}(0, \mathbf{I})$ to the data endpoint through a learned velocity field $v_\theta(x_n, n, o_t, l)$ over flow time $n \in [0, 1]$. We adopt the linear interpolation path

$$x_n = (1 - n)x_0 + nx_1, \qquad x_1 \sim \mathcal{N}(0, \mathbf{I}), \qquad n \sim \mathcal{U}(0, 1), \tag{3}$$

for which the target velocity is

$$u_n = x_1 - x_0. \tag{4}$$

The action chunk is obtained by solving

$$\frac{\mathrm{d}x_n}{\mathrm{d}n} = v_\theta(x_n, n, o_t, l) \tag{5}$$

from $n = 1$ to $n = 0$. The velocity field is pretrained on a dataset of expert demonstrations $\mathcal{D}_{\text{exp}}$ using the standard conditional flow-matching objective

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{(o_t, x_0, l) \sim \mathcal{D}_{\text{exp}},\, x_1 \sim \mathcal{N}(0, \mathbf{I}),\, n \sim \mathcal{U}(0,1)} \left[\| v_\theta(x_n, n, o_t, l) - u_n \|_2^2 \right]. \tag{6}$$

3.2 Execution-Context Matching

Exact state matches are rare across independently collected rollouts, even when the robot is attempting the same local behavior. We therefore represent the state associated with each action chunk by an *execution context* that captures both task stage and robot configuration. Clustering these contexts provides cross-trajectory correspondences that are used by the next module to identify locally supported corrections.

Progress-aware context representation. Task progress indicates what the robot is attempting to accomplish at a given point in a rollout. We estimate it using the pretrained Robo-Dopamine 2.0 General Reward Model (GRM) (Xu et al., 2026) in a zero-shot manner, without task-specific fine-tuning. We denote the progress estimate by $R(o_t, l, \mathcal{H}_t)$, where $\mathcal{H}_t$ is a chronologically ordered

sequence of visual observations sampled from the rollout before step t. Since raw GRM predictions may fluctuate across nearby chunks, we smooth the progress sequence using a box filter:

$$\bar{p}_t = \frac{1}{|\mathcal{J}_t|} \sum_{j \in \mathcal{J}_t} R(o_j, l, \mathcal{H}_j), \qquad \mathcal{J}_t = \{j : \max(0, t-W) \le j \le \min(T-1, t+W)\}, \tag{7}$$

where $W > 0$ denotes the half-window size.

We combine the smoothed progress estimate with the proprioceptive state to represent the execution context of the action chunk:

$$f_t = [\, \tilde{q}_t;\ \beta\, \bar{p}_t \,], \tag{8}$$

where $\tilde{q}_t$ denotes normalized proprioception and $\beta > 0$ balances the relative scales of proprioception and task progress. Task progress indicates the current stage of execution, while proprioception distinguishes different robot configurations within that stage. This is a deliberately low-dimensional matching key: it omits image-level object pose, contact, and occlusion state, so we do not assume that it is sufficient for every corrective decision.

Context matching. For each task, we cluster the execution contexts of all action chunks in $\mathcal{D}$ using HDBSCAN (McInnes & Healy, 2017), yielding clusters $\{\mathcal{C}_c\}$. Chunks assigned to the same cluster have comparable estimated task progress and robot configurations, but are not assumed to share identical visual, object, or contact states. HDBSCAN outliers remain unmatched. This stage does not yet determine whether an action is desirable; it defines the local cross-trajectory support within which Quality-Guided Action Redirection evaluates actions and constructs corrective references.

3.3 Quality-Guided Action Redirection

Given the execution-context clusters, we next determine which action chunks should be reinforced, suppressed, or redirected. We first estimate chunk-level quality from progress change and trajectory outcome, and then use the resulting labels to construct corrective targets from higher-quality actions in matched contexts.

Chunk-level quality estimation. Using the smoothed progress sequence from Eq. equation 7, we combine local progress change with the trajectory outcome to define the signed chunk-level quality score

$$\hat{A}_t = \bar{p}_{\min(t+W, T-1)} - \bar{p}_{\max(t-W, 0)} + b \cdot \big(2\, \mathbb{1}[y_\tau{=}1] - 1\big), \tag{9}$$

where $b > 0$ controls the contribution of the trajectory-level outcome. The progress-difference term captures whether task progress increases around the current chunk, while the outcome term provides coarse episode-level information when local progress is ambiguous. This construction is a retrospective signed chunk-quality score: all chunks in an episode inherit the same outcome term, so neutral or necessary actions can receive an imperfect sign. We therefore use $\hat{A}_t$ for relative weighting and target assignment rather than interpreting it as a policy advantage or an identified counterfactual effect of the chunk.

Chunks with $\hat{A}_t > 0$ are labeled positive, while chunks with $\hat{A}_t < 0$ are labeled negative. Chunks with $\hat{A}_t = 0$ are not assigned corrective targets and are handled by the soft weighting defined below.

Corrective-target construction. For each matched context cluster $\mathcal{C}_c$, let

$$\mathcal{C}_c^+ = \{\, i \in \mathcal{C}_c : \hat{A}_i > 0 \,\}$$

denote its positive subset. For a negative chunk a_t assigned to $\mathcal{C}_c$, if $\mathcal{C}_c^+$ is non-empty, we construct a corrective target as a quality-weighted centroid of the positive actions in that cluster:

$$\alpha_i = \frac{\exp(\hat{A}_i/\kappa)}{\sum_{j \in \mathcal{C}_c^+} \exp(\hat{A}_j/\kappa)}, \qquad a_t^\star = \sum_{i \in \mathcal{C}_c^+} \alpha_i a_i, \tag{10}$$

where $\kappa > 0$ controls how strongly higher-quality actions are emphasized. The weighted centroid is a locally supported corrective reference, not a guaranteed counterfactual action for every member of a multimodal cluster. A negative chunk is treated as *uncorrectable* if it is an HDBSCAN outlier

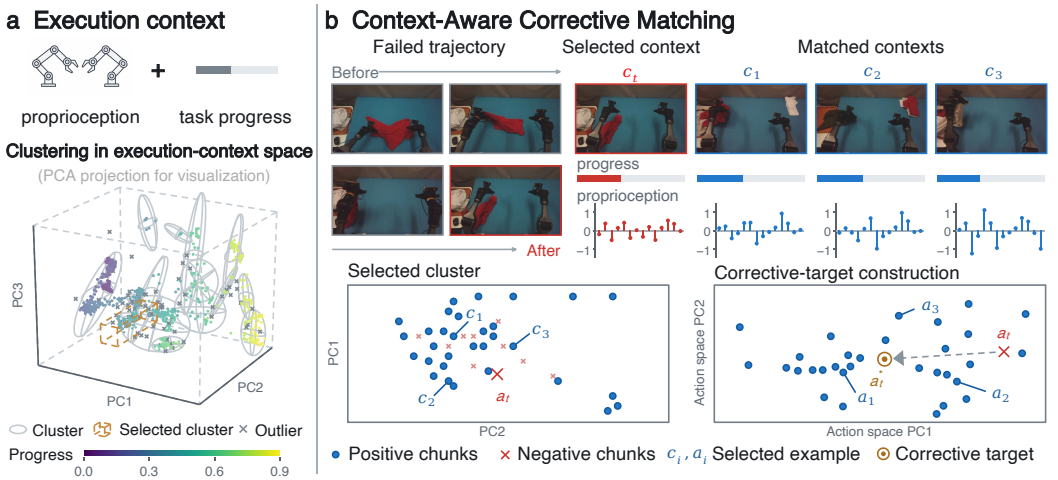


Figure 3: **Context-aware corrective matching and target construction.** **(a)** Each action chunk is represented by its execution context $(\tilde{q}_t, \bar{p}_t)$, combining robot proprioception and task progress, and grouped with HDBSCAN. The 3D view is a PCA projection for visualization; clustering is performed in the full execution-context space, with outliers left unmatched. **(b)** For a selected negative context c_t, RedFlow retrieves positive contexts from the same context cluster C. The real-robot examples c_1, c_2, c_3 illustrate representative matches with comparable progress and proprioception despite different visual observations. Their cluster-level positive support is then aggregated in action space according to chunk quality to construct the corrective target $a_t^\star$, which provides the direction for redirecting the negative chunk. The labeled a_1, a_2, a_3 are representative examples, not the complete set used to form $a_t^\star$.

or if $\mathcal{C}_c^+ = \emptyset$; such chunks receive suppression but no target-guided correction. Appendix E reports target coverage, action dispersion, and alternative target constructions.

With quality labels and corrective targets in place, we modify the standard flow-matching objective, which otherwise imitates every action chunk in the same way. Our objective follows three principles: reinforce high-quality actions, suppress low-quality actions, and redirect correctable negative actions toward their matched positive targets.

Quality-weighted attraction. We first use the chunk-level quality score $\hat{A}_t$ to control how strongly each observed action should be imitated:

$$w_t = \sigma(\hat{A}_t / T_w) \in (0, 1), \tag{11}$$

where $T_w > 0$ is a temperature. Higher-quality chunks receive larger weights, while lower-quality chunks receive smaller weights. We then weight the standard flow-matching loss as

$$\mathcal{L}_{\text{att}} = w_t \left\| v_\theta(x_n, n, o_t, l) - u_n \right\|^2. \tag{12}$$

This term preserves useful behaviors in the offline buffer while reducing the imitation strength of low-quality chunks.

Failure suppression. Simply assigning a small imitation weight to a negative chunk does not explicitly discourage the policy from predicting that action. We therefore introduce a suppression term that pushes the predicted action away from negative chunks when they are too close.

We first reconstruct the predicted clean action endpoint:

$$\hat{x}_0 = x_n - n\, v_\theta(x_n, n, o_t, l), \qquad e_t = \|\hat{x}_0 - a_t\|^2. \tag{13}$$

The suppression loss is

$$\mathcal{L}_{\text{sup}} = \mathbb{1}[\hat{A}_t < 0]\, \lambda_{\text{sup}} (1 - w_t) \max(0, m - e_t), \tag{14}$$

where $\lambda_{\text{sup}} > 0$ controls the suppression strength and m is an adaptive margin. We update this margin once per logical optimizer update, rather than once per epoch or in the middle of a gradient-accumulation cycle. Specifically, let $\mathcal{B}_k$ denote the union of all micro-batches that contribute to

optimizer update k, and let m_k be held fixed while the corresponding gradients are accumulated. After the policy optimizer step, we compute the detached reconstruction-error statistic

$$\bar{e}_k = \frac{1}{|\mathcal{B}_k|} \sum_{t \in \mathcal{B}_k} \mathrm{stopgrad}(e_t), \qquad m_{k+1} = (1-\rho) m_k + \rho\, s_m \bar{e}_k, \tag{15}$$

where $\rho \in (0, 1]$ is the running-average coefficient and s_m is the margin scale. Thus every micro-batch in one logical update sees the same stop-gradient margin, while the margin can adapt after each optimizer step. The numerical values of ρ, s_m, and the initialization of m_0 are implementation hyperparameters; the update frequency and aggregation rule are specified here and summarized in Appendix C. The indicator $\mathbb{1}[\hat{A}_t < 0]$ ensures that suppression is applied only to negative chunks. For these chunks, the loss pushes the predicted action away from the observed action when their squared distance is below m. Once the distance exceeds m, the suppression term becomes inactive.

Target-guided correction. Suppression tells the policy which action to avoid, but not what it should do instead. For negative chunks with assigned corrective targets, we therefore introduce a correction term:

$$\mathcal{L}_{\mathrm{cor}} = c_t\, \lambda_{\mathrm{cor}} (1 - w_t) \|\hat{x}_0 - a_t^\star\|^2, \tag{16}$$

where $\lambda_{\mathrm{cor}} > 0$ controls the correction strength and $c_t \in \{0, 1\}$ indicates whether a negative chunk has an assigned corrective target. Specifically, $c_t = 1$ only when $\hat{A}_t < 0$ and $a_t^\star$ is available; otherwise, $c_t = 0$. Thus, correction is applied only to correctable negative chunks, while negative chunks without corrective targets receive suppression alone.

The correction term guides the predicted action toward the locally supported action reference constructed from positive chunks in the matched execution context. Because $\hat{x}_0 = x_n - n v_\theta$, the endpoint terms have flow-time-dependent gradients with respect to the velocity field; the coefficients λ_{sup} and λ_{cor} therefore do not imply uniform redirection over n. We make this scaling explicit in Appendix A.3. The target is used as an additional corrective signal during post-training rather than as a direct replacement for the original action.

The local interaction between suppression and correction is characterized by the following result.

Theorem 1 (Bounded endpoint redirection)**.** *Consider a predicted clean action endpoint $h \in \mathcal{A}$, a negative action chunk $a^- \in \mathcal{A}$, and a corrective target $b \in \mathcal{A}$. If $b \neq a^-$, $m > 0$, and $\lambda_{\mathrm{sup}} \geq \lambda_{\mathrm{cor}} > 0$, define*

$$\phi(h) = \lambda_{\mathrm{cor}} \|h - b\|^2 + \lambda_{\mathrm{sup}} [m - \|h - a^-\|^2]_+, \tag{17}$$

where $[r]_+ = \max(r, 0)$. The minimizer of ϕ is the point closest to the corrective target b while remaining outside the margin region around a^-. Equivalently, it is the projection of b onto $\{h : \|h - a^-\|^2 \geq m\}$. The proof is given in Appendix A.

This theorem is a local statement for the fixed-n suppression–correction subsystem, not a characterization of the full neural-network objective, which also includes quality-weighted attraction.

Total objective. Combining the three terms, the final objective is

$$\mathcal{L} = \mathbb{E}_{(o_t, a_t, l) \sim \mathcal{D},\, x_1 \sim \mathcal{N}(0, \mathbf{I}),\, n \sim \mathcal{U}(0,1)} \left[\mathcal{L}_{\mathrm{att}} + \mathcal{L}_{\mathrm{sup}} + \mathcal{L}_{\mathrm{cor}} \right]. \tag{18}$$

The three terms have complementary roles: $\mathcal{L}_{\mathrm{att}}$ preserves high-quality observed behavior, $\mathcal{L}_{\mathrm{sup}}$ discourages negative actions, and $\mathcal{L}_{\mathrm{cor}}$ guides selected negative chunks toward locally supported corrective references.

RedFlow follows a single-iteration offline procedure summarized in Algorithm 1 of Appendix B. We first collect a fixed rollout buffer using the pretrained flow-matching VLA policy, construct execution-context clusters as described in Section 3.2, and then estimate chunk quality, construct corrective targets, and optimize the policy using the objective above.

4 Experiments

We evaluate RedFlow against offline and online policy-improvement baselines, ablate its components, and test it on real-robot manipulation tasks.

Table 1: Success rates (%) on the LIBERO benchmark across four task suites.

Method	Spatial	Object	Goal	Long	Avg.
Base Policy	63.6	61.6	48.6	50.8	56.2
AWR	71.2	66.8	57.8	53.4	62.3
DPO	66.8	64.8	51.8	51.2	58.7
CQL	70.4	63.0	52.6	52.0	59.5
IQL	71.6	63.4	55.4	52.8	60.8
RedFlow (Ours)	**75.8**	**70.4**	**71.2**	**55.2**	**68.2**

Table 2: Ablation studies on three LIBERO suites. Each number reports a success rate (%).

Method	Spatial	Object	Goal	Avg.
Training data composition				
w/o low-quality rollouts	71.4	67.4	65.2	68.0
w/o high-quality rollouts	64.4	65.8	57.0	62.4
Execution-Context Matching				
w/o task progress	50.8	46.2	33.6	43.5
w/o proprioceptive state	62.4	60.2	52.0	58.2
Quality-Guided Action Redirection				
w/o correctability filtering	66.4	65.8	50.8	61.0
w/o $\mathcal{L}_{\text{cor}}$	70.4	64.4	55.6	63.5
w/o $\mathcal{L}_{\text{sup}}$	70.8	66.8	68.8	68.8
w/o $\mathcal{L}_{\text{sup}}$ & $\mathcal{L}_{\text{cor}}$	63.8	62.8	70.4	65.7
RedFlow (Ours)	**75.8**	**70.4**	**71.2**	**72.5**

4.1 Experimental Setup

Simulation Experiments. We use the four 10-task LIBERO suites (Liu et al., 2023) and a π_0 policy initialized from π_{RL} (Black et al., 2024a; Chen et al., 2026a). Each suite uses a fixed buffer of 1,536 mixed-quality base-policy rollouts; the SFT data used to initialize the base policy are excluded from post-training. Our simulation training implementation builds on RLinf (Yu et al., 2026a). All offline methods share the policy, buffer, Robo-Dopamine 2.0 progress model (Xu et al., 2026), and training protocol and are evaluated over 500 episodes. Appendix C provides full details.

Real-Robot Experiments. We evaluate clothes folding, object sweeping, and table cleaning on a dual-arm Agilex Cobot Magic using task-specific π_0 base policies and fixed mixed-quality rollout buffers. We report 100 trials per task; hardware, data sizes, task visualizations, and training details appear in Appendix D.

4.2 Main Results

Table 1 compares RedFlow with AWR, DPO, CQL, and IQL (Peng et al., 2019; Rafailov et al., 2023; Kumar et al., 2020; Kostrikov et al., 2022). All methods share the base policy, buffer, action chunks, and progress/outcome supervision; Appendix C details their flow-policy implementations. RedFlow obtains the best average result, improving the base policy from 56.2% to **68.2%**; its largest gain is on Goal (48.6% to 71.2%). RedFlow exceeds the reported reweighting, preference, and value-based baselines, although close per-suite margins should be interpreted in light of their flow-policy adaptations. Wilson 95% confidence intervals over evaluation episodes are reported in Appendix C.

4.3 Ablation Studies

Table 2 evaluates data composition, matching context, and loss terms on Spatial, Object, and Goal. The full mixed-quality buffer is important: removing its low- or high-quality subset lowers the average from 72.5% to 68.0% or 62.4%. Progress and proprioception are complementary matching cues (43.5% and 58.2% when removed). Within Quality-Guided Action Redirection, assigning targets without positive support yields 61.0%, while removing correction, suppression, or both gives 63.5%, 68.8%, and 65.7%, respectively. The latter isolates a 6.8-point gain from explicit redirection beyond quality-weighted attraction. The larger drop without correction further shows that suppression benefits from a locally supported target direction rather than repulsion alone. Appendix E further shows that combining progress and outcome reaches 68.2% across all four suites, compared with 62.2% for progress only and 54.1% for outcome only; shuffling trajectory outcomes reduces performance to 38.2%. The appendix also reports target coverage, distances, and alternative target constructions.

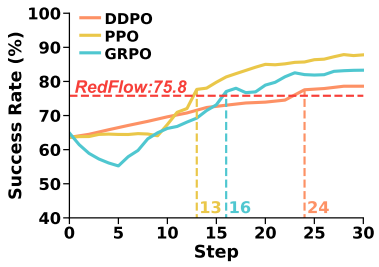


Figure 4: Threshold-matched sample-efficiency comparison on LIBERO-Spatial against on-policy RL baselines. Each step corresponds to 1,024 fresh rollouts.

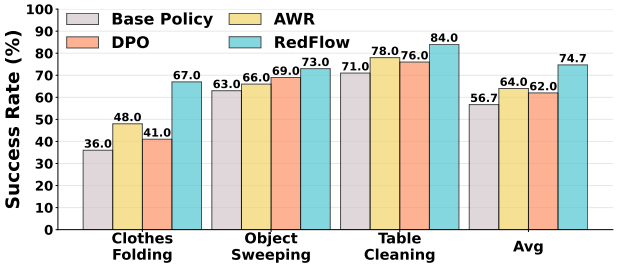


Figure 5: Success rates on real-robot tasks.

4.4 Threshold-Matched Sample Efficiency

On LIBERO-Spatial, RedFlow reaches 75.8% using 1,536 fixed rollouts and no online interaction during policy optimization (Figure 4). PPO, GRPO, and DDPO (Schulman et al., 2017; Shao et al., 2024; Black et al., 2024b) first reach this threshold after 13, 16, and 24 steps of 1,024 fresh rollouts, respectively (13,312, 16,384, and 24,576 rollouts), or 8.7–16× as many. This is a threshold-matched comparison of additional post-training interaction, not an end-to-end data-efficiency claim.

4.5 Real-Robot Results

RedFlow improves all three real-robot tasks, averaging 74.7% versus 56.7% for the base policy (Figure 5); clothes folding shows the largest gain, from 36.0% to 67.0%. These results are limited to the evaluated tasks and reset distributions. Exact success counts and Wilson 95% confidence intervals are reported in Appendix D.

Qualitative Recovery Behavior. In Figure 6, the base policy repeats an unreachable right-arm grasp, whereas RedFlow uses the left arm to return the cloth to a reachable configuration. This example is consistent with context-matched redirection toward a locally supported reference, but serves only as an illustration rather than as standalone validation of the mechanism.

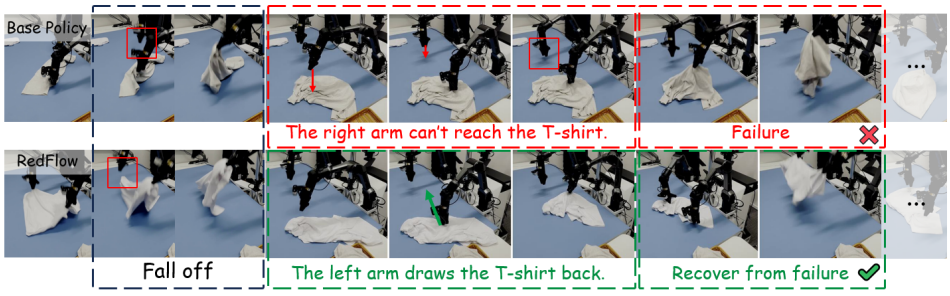


Figure 6: Qualitative recovery behavior on clothes folding. The base policy repeatedly attempts an unreachable right-arm grasp, while RedFlow uses the left arm to pull the cloth back into a reachable configuration before continuing the task.

5 Conclusion

We presented RedFlow, which constructs locally supported corrective targets from mixed-quality rollouts for offline post-training of flow-matching VLAs. It improves the LIBERO base-policy average from 56.2% to 68.2% and the real-robot average from 56.7% to 74.7%. At the matched LIBERO-Spatial threshold, the evaluated online methods require 8.7–16× as many additional rollouts. These results support corrective reuse of offline experience; future work should improve target uncertainty estimation and test broader, longer-horizon settings.

AI USE STATEMENT

In this work, we used chat GPT-6 solely to improve the grammar, clarity, and readability of the manuscript. We did not use generative AI tools for any research tasks requiring disclosure under the ICLR 2027 AI Policy for Authors. The authors reviewed all AI-assisted edits and checked that they preserved the intended technical meaning. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops and evaluates RedFlow using the publicly available LIBERO benchmark, newly collected simulation rollouts, and newly collected real-robot demonstrations and rollout data. We used the benchmark data in accordance with its applicable license and terms of use. The real-robot experiments involved physical manipulation tasks but did not recruit human participants or collect new personal data. Our evaluation is limited to the studied tasks and reset distributions and does not establish suitability for deployment in high-stakes applications. Potential ethical concerns in downstream use should be assessed in the context of the specific application.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe the proposed method in Section 3. The datasets, experimental protocols, and evaluation metrics are detailed in Section 4. Additional implementation details, including data preprocessing, hyperparameter settings, and computational resources, are provided in Appendix B. Source code and instructions for reproducing the experiments are included in the supplementary materials.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024a.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning, 2024b. URL https://arxiv.org/abs/2305.13301.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2026a. URL https://arxiv.org/abs/2510.25889.

Yiye Chen, Yanan Jian, Xiaoyi Dong, Shuxin Cao, Jing Wu, Patricio Vela, Benjamin E. Lundell, and Dongdong Chen. Vista: Enhancing visual conditioning via track-following preference optimization in vision-language-action models. *arXiv preprint arXiv:2602.05049*, 2026b.

Charles A. Hepburn and Giovanni Montana. Model-based trajectory stitching for improved offline reinforcement learning, 2022.

Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. Co-rft: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *arXiv preprint arXiv:2508.02219*, 2025.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili

Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL https://arxiv.org/abs/1805.00909.

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.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. Simplevla-rl: Scaling vla training via reinforcement learning, 2025. URL https://arxiv.org/abs/2509.09674.

Zhuohang Li, Liqun Huang, Wei Xu, Zhengming Zhu, Nie Lin, Xiao Ma, Xinjun Sheng, and Ruoshi Wen. Hand-in-the-loop: Improving vla policies for dexterous manipulation via seamless hand-arm intervention. *arXiv preprint arXiv:2605.15157*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18719.

Leland McInnes and John Healy. Accelerated hierarchical density based clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 33–42, 2017. doi: 10.1109/ICDMW.2017.12.

Yunpeng Mei, Jiakai He, Hongjie Cao, Chenyu Wang, Xiaowen Zhu, Yihan Zhou, Jiamin Wang, Chenbo Xin, Peng Cheng, Yuxuan Yang, Yijie Wang, Xinhu Zheng, Gao Huang, Jie Chen, and Gang Wang. Potential-guided flow matching for vision-language-action policy improvement, 2026.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019. URL https://arxiv.org/abs/1910.00177.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive rl finetuning, 2026.

Siting Wang, Xiaofeng Wang, Zheng Zhu, Minnan Pei, Xinyu Cui, Cheng Deng, Jian Zhao, Guan Huang, Haifeng Zhang, and Jun Wang. π-stepnft: Wider space needs finer steps in online rl for flow-based vlas, 2026a. URL https://arxiv.org/abs/2603.02083.

Ziqian Wang, Rui Zhang, Yitian Liu, Xingjian Mao, Minqian Wang, and Yao Mu. Q-vgm: Q-guided value-gradient matching for offline-to-online rl of flow-matching vla policies, 2026b.

Yihao Wu, He Zhang, Junbo Tan, Xueqian Wang, and Zhengyou Zhang. Flowpro: Reward-free reinforced fine-tuning of flow-matching vlas via proximalized preference optimization, 2026.

Yijie Xu, Haopeng Jin, Run Zhou, Shengbang Liu, Sixiang Chen, Hongyang Cheng, Sicheng Hu, Peterson Co, Jinwen Luo, Huajie Tan, and Shanghang Zhang. Robo-Dopamine 2.0: History-conditioned and OOD-aware process reward modeling for robotic manipulation, 2026. URL https://arxiv.org/abs/2608.15680.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. {RLinf}: Flexible and efficient {Large-Scale} reinforcement learning via {Macro-to-Micro} flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 829–846, 2026a.

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

Hongyin Zhang, Shiyuan Zhang, Junxi Jin, Qixin Zeng, Yifan Qiao, Hongchao Lu, and Donglin Wang. Balancing signal and variance: Adaptive offline rl post-training for vla flow models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18755–18763, 2026.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

A Analysis of Quality-Guided Action Redirection

This appendix provides a local analysis of the main components of Quality-Guided Action Redirection. Our goal is not to characterize the global optimization dynamics of the neural policy, but to

clarify three properties of the proposed objective: (i) the interpretation of the cluster-derived corrective target, (ii) the interaction between failure suppression and target-guided correction, and (iii) the relation between endpoint correction and flow-matching velocity regression.

Throughout, $\mathcal{A} = \mathbb{R}^{K\times D}$ denotes the action-chunk space and $\|\cdot\|$ denotes the Frobenius norm.

A.1 Corrective Target as a Weighted Local Mean

Recall that for a cluster $\mathcal{C}_c$ with positive subset $\mathcal{C}_c^+$, RedFlow constructs the corrective target

$$a_t^\star = \sum_{i\in\mathcal{C}_c^+} \alpha_i a_i, \qquad \alpha_i = \frac{\exp(\hat{A}_i/\kappa)}{\sum_{j\in\mathcal{C}_c^+}\exp(\hat{A}_j/\kappa)}. \tag{19}$$

The weights satisfy $\alpha_i > 0$ and $\sum_{i\in\mathcal{C}_c^+}\alpha_i = 1$.

Under the squared Euclidean metric, $a_t^\star$ is the unique minimizer of

$$\min_{a\in\mathcal{A}} \sum_{i\in\mathcal{C}_c^+} \alpha_i \|a - a_i\|^2. \tag{20}$$

Indeed, differentiating Eq. equation 20 with respect to a gives

$$\sum_{i\in\mathcal{C}_c^+} \alpha_i (a - a_i) = 0, \tag{21}$$

whose unique solution is $a = \sum_i \alpha_i a_i = a_t^\star$.

Thus, the corrective target summarizes the positive actions observed within the matched execution-context cluster, with greater emphasis on higher-quality chunks. Importantly, $a_t^\star$ is not interpreted as the exact counterfactual action that must replace a_t. It is used only as a local corrective reference during post-training.

A.2 Bounded Interaction of Suppression and Correction

We next analyze the interaction between failure suppression and target-guided correction for a single correctable negative chunk. This analysis concerns only the corrective subsystem of the objective and does not constitute a convergence guarantee for the full neural-network training procedure.

Consider a predicted clean action endpoint $h \in \mathcal{A}$, a negative action chunk $a^- \in \mathcal{A}$, and a corrective target $b \in \mathcal{A}$. Define

$$\phi(h) = \lambda_{\text{cor}}\|h - b\|^2 + \lambda_{\text{sup}}[m - \|h - a^-\|^2]_+, \tag{22}$$

where $[r]_+ = \max(r, 0)$, $m > 0$, and $\lambda_{\text{sup}}, \lambda_{\text{cor}} > 0$.

Theorem 2 (Bounded corrective redirection)**.** *Assume $b \neq a^-$ and $\lambda_{\text{sup}} \geq \lambda_{\text{cor}} > 0$. The minimizer of Eq. equation 22 is*

$$h^\star = \begin{cases} b, & \|b - a^-\|^2 \geq m, \\ a^- + \sqrt{m}\,\dfrac{b - a^-}{\|b - a^-\|}, & \|b - a^-\|^2 < m. \end{cases} \tag{23}$$

Equivalently, $h^\star$ is the solution of

$$\min_{h\in\mathcal{A}} \|h - b\|^2 \qquad \text{s.t.} \qquad \|h - a^-\|^2 \geq m. \tag{24}$$

Proof. If $\|b - a^-\|^2 \geq m$, the hinge term is inactive at $h = b$, and $h = b$ minimizes both terms.

Now suppose $d = \|b - a^-\| < \sqrt{m}$. For a fixed radius $r = \|h - a^-\|$, the attraction term is minimized when h lies on the ray from a^- toward b:

$$h = a^- + r\frac{b - a^-}{d}. \tag{25}$$

The objective inside the margin therefore reduces to

$$\psi(r) = \lambda_{\text{cor}}(d - r)^2 + \lambda_{\text{sup}}(m - r^2), \qquad 0 \le r < \sqrt{m}. \tag{26}$$

Its derivative is

$$\psi'(r) = 2(\lambda_{\text{cor}} - \lambda_{\text{sup}})r - 2\lambda_{\text{cor}}d. \tag{27}$$

Since $\lambda_{\text{sup}} \ge \lambda_{\text{cor}} > 0$ and $d > 0$, we have $\psi'(r) < 0$ for $0 \le r < \sqrt{m}$. Hence the objective decreases until the boundary $r = \sqrt{m}$. Outside the margin, the hinge term vanishes, and the closest feasible point to b is precisely Eq. equation 23. □

The theorem shows that suppression is finite-range: it pushes the prediction away from a negative action only while the prediction lies within the margin. The correction term simultaneously determines the direction in which the prediction is moved. Therefore, suppression does not induce unrestricted repulsion from the negative action.

For a correctable negative chunk in RedFlow, the corresponding subsystem is

$$\begin{aligned}\mathcal{L}_{\text{local}} = \lambda_{\text{sup}}(1 - w_t)[m - \|\hat{x}_0 - a_t\|^2]_+ \\ + \lambda_{\text{cor}}(1 - w_t)\|\hat{x}_0 - a_t^\star\|^2.\end{aligned} \tag{28}$$

For any fixed $w_t < 1$, the common positive factor $(1-w_t)$ does not change the minimizer. Therefore, Theorem 2 directly characterizes the interaction between the suppression and correction terms when $a^- = a_t$ and $b = a_t^\star$.

The full RedFlow objective additionally contains the quality-weighted flow-matching attraction term. Its role is to preserve supervision from observed behavior data; the theorem above is intended only to characterize the local geometry of the corrective subsystem.

A.3 Endpoint Correction in Flow Matching

We finally show how the endpoint-space correction used by RedFlow relates to the velocity parameterization of flow matching.

We use the linear interpolation

$$x_n = (1 - n)x_0 + nx_1, \qquad x_1 \sim \mathcal{N}(0, \mathbf{I}), \qquad n \in (0, 1], \tag{29}$$

with target velocity

$$u_n = x_1 - x_0. \tag{30}$$

Given a predicted velocity $v_\theta(x_n, n, z)$, where z denotes the policy conditioning variables, the corresponding clean endpoint estimate is

$$\hat{x}_0 = x_n - nv_\theta(x_n, n, z). \tag{31}$$

For any desired endpoint $y \in \mathcal{A}$, define the velocity target

$$u_n^y(x_n, z) = \frac{x_n - y}{n}. \tag{32}$$

Proposition 3 (Endpoint and velocity regression)**.** *For any fixed $n > 0$,*

$$\|\hat{x}_0 - y\|^2 = n^2 \left\|v_\theta(x_n, n, z) - u_n^y(x_n, z)\right\|^2. \tag{33}$$

Therefore, endpoint regression toward y and velocity regression toward u_n^y have the same minimizer for fixed n.

Proof. Using Eq. equation 31,

$$\hat{x}_0 - y = x_n - y - nv_\theta(x_n, n, z) \tag{34}$$

$$= n\left(\frac{x_n - y}{n} - v_\theta(x_n, n, z)\right). \tag{35}$$

Taking the squared norm gives Eq. equation 33. □

This relation explains why corrective supervision can be introduced directly in the predicted endpoint space while remaining compatible with the flow-matching parameterization. Choosing $y = a_t$ recovers attraction toward an observed action chunk, whereas choosing $y = a_t^\star$ gives attraction toward the matched corrective target.

Flow-time gradient scaling. The endpoint equivalence above is a statement about minimizers for a fixed n; it does not imply that the corresponding parameter gradients have the same magnitude across flow time. For an endpoint target y, the chain rule gives

$$\nabla_{v_\theta} \|\hat{x}_0 - y\|^2 = -2n(\hat{x}_0 - y), \tag{36}$$

where the gradient is taken with respect to the predicted velocity at the current (x_n, n, z). Consequently, the correction gradient is

$$\nabla_{v_\theta} \mathcal{L}_{\text{cor}} = -2n\, c_t \lambda_{\text{cor}} (1 - w_t)(\hat{x}_0 - a_t^\star). \tag{37}$$

When the suppression hinge is active, its gradient is

$$\nabla_{v_\theta} \mathcal{L}_{\text{sup}} = 2n\, \mathbb{1}[\hat{A}_t < 0] \lambda_{\text{sup}} (1 - w_t)(\hat{x}_0 - a_t). \tag{38}$$

Both endpoint gradients therefore vanish linearly as $n \to 0$, whereas the velocity-regression term has no corresponding explicit multiplicative factor of n. The fixed coefficients in the main objective should thus be read as empirical loss weights, not as a guarantee of uniform endpoint redirection over flow time. Reweighting by a function of n or changing the sampling density of n is a distinct design choice that requires separate empirical validation.

The suppression term is also defined in endpoint space:

$$\mathcal{L}_{\text{sup}} = \lambda_{\text{sup}} (1 - w_t)[m - \|\hat{x}_0 - a_t\|^2]_+. \tag{39}$$

Its gradient with respect to $\hat{x}_0$ is nonzero only when $\|\hat{x}_0 - a_t\|^2 < m$:

$$-\nabla_{\hat{x}_0} [m - \|\hat{x}_0 - a_t\|^2]_+ = 2\, \mathbb{1}[\|\hat{x}_0 - a_t\|^2 < m](\hat{x}_0 - a_t). \tag{40}$$

Thus, the suppression term acts only within a finite neighborhood of a negative chunk, while the correction term provides a direction toward its matched positive reference.

A.4 Summary

The analysis above establishes three local properties of RedFlow. First, the corrective target is the quality-weighted squared-distance mean of positive actions within the matched execution context. Second, the suppression–correction subsystem produces bounded repulsion from a negative action while directing the prediction toward its corrective target. Third, endpoint-space correction has a direct realization under the flow-matching velocity parameterization.

These results characterize the local geometry of the proposed objective. They do not imply global convergence of the full neural-network optimization procedure, nor do they require interpreting RedFlow as an optimal-transport algorithm.

B Training Details

Overall training pipeline. RedFlow follows a single-iteration offline post-training pipeline for flow-matching VLA policies. We first obtain a task-specific base policy by fine-tuning π_0 on expert demonstrations. To fine-tune π_0 and obtain the base policy, we use eight H20 GPUs. We then use the base policy to collect a fixed offline buffer containing 1,536 mixed-quality rollout trajectories per suite. Given the frozen buffer, we estimate task progress with the pretrained Robo-Dopamine 2.0 GRM, form progress–state execution-context clusters, estimate chunk-level quality, construct corrective targets within matched clusters, and finally update the policy using Quality-Guided Action Redirection.

GRM usage. We use the pretrained Robo-Dopamine 2.0 GRM in a zero-shot manner. Its parameters remain frozen during all simulation and real-robot experiments, and no rollout data collected in this work are used to fine-tune the GRM.

Execution-context matching. For both simulation and real-robot experiments, we use the pretrained Robo-Dopamine 2.0 GRM (Xu et al., 2026) to estimate task progress and apply the execution-context representation described in Section 3.2. Progress estimates are smoothed over temporal chunks and combined with proprioceptive states for HDBSCAN (McInnes & Healy, 2017) clustering in the normalized progress–state space. Detailed HDBSCAN and progress-estimation hyperparameters are provided separately for simulation and real-robot experiments in Tables 8 and 10.

Chunk quality and corrective-target assignment. Quality-Guided Action Redirection combines progress changes with trajectory outcomes to obtain the signed chunk-quality score $\hat{A}_t$ and constructs corrective targets from co-clustered positive chunks when available. Negative chunks without positive support in their matched cluster receive suppression only.

Offline policy optimization. The policy is optimized with Quality-Guided Action Redirection in Eq. equation 18, which combines quality-weighted attraction, failure suppression, and target-guided correction. The adaptive margin is held fixed over each logical optimizer update and updated once after the optimizer step according to Eq. equation 15; with gradient accumulation, the statistic is computed over the union of the contributing micro-batches. The loss coefficients and optimization hyperparameters are reported in the corresponding simulation and real-robot implementation tables.

Algorithm. The full training procedure is summarized in Algorithm 1.

Algorithm 1 RedFlow training pipeline

Require: A pretrained policy π_θ, a GRM R, a task instruction l, and the number of epochs E

// Phase 1: Data collection

$\mathcal{D}_{\text{roll}} \leftarrow \{\tau_i\}_{i=1}^{N}$ via rollouts of π_θ

$\mathcal{D} \leftarrow \mathcal{D}_{\text{roll}}$

// Phase 2: Execution-Context Matching

for $a_t \in \mathcal{D}$ **do**

  $\mathcal{J}_t \leftarrow \{j : \max(0, t-W) \le j \le \min(T-1, t+W)\}$

  $\bar{p}_t \leftarrow \frac{1}{|\mathcal{J}_t|} \sum_{j \in \mathcal{J}_t} R(o_j, l, \mathcal{H}_j)$

  $f_t \leftarrow [\tilde{q}_t; \beta \bar{p}_t]$

end for

$\{\mathcal{C}_c\} \leftarrow \text{HDBSCAN}(\{f_t\})$

// Phase 3: Quality-Guided Action Redirection

for $a_t \in \mathcal{D}$ **do**

  $\hat{A}_t \leftarrow \bar{p}_{\min(t+W, T-1)} - \bar{p}_{\max(t-W, 0)} + b(2\mathbb{1}[y_\tau = 1] - 1)$

  $d_t \leftarrow \mathbb{1}[\hat{A}_t < 0]$

  $c_t \leftarrow 0$

end for

$\mathcal{C}_c^+ \leftarrow \{i \in \mathcal{C}_c : \hat{A}_i > 0\}$ for each cluster $\mathcal{C}_c$

for a_t with $d_t = 1$ **do**

  Let $\mathcal{C}_c$ be the cluster assigned to a_t

  if a_t is not an outlier and $\mathcal{C}_c^+ \neq \emptyset$ **then**

    $\alpha_i \leftarrow \text{softmax}(\hat{A}_i / \kappa)_{i \in \mathcal{C}_c^+}$

    $a_t^\star \leftarrow \sum_{i \in \mathcal{C}_c^+} \alpha_i a_i$

    $c_t \leftarrow 1$

  end if

end for

for $e = 1, \ldots, E$ **do**

  for each minibatch (o_t, a_t, l) from $\mathcal{D}$ **do**

    $x_1 \sim \mathcal{N}(0, \mathbf{I}), \quad n \sim \mathcal{U}(0, 1)$

    $x_n \leftarrow (1-n)a_t + n x_1, \quad u_n \leftarrow x_1 - a_t$

    $\hat{x}_0 \leftarrow x_n - n v_\theta(x_n, n, o_t, l)$

    $w_t \leftarrow \sigma(\hat{A}_t / T_w)$

    Compute $\mathcal{L}_{\text{att}}$ using Eq. equation 12

    Compute $\mathcal{L}_{\text{sup}}$ using Eq. equation 14 with mask d_t

    Compute $\mathcal{L}_{\text{cor}}$ using Eq. equation 16 with indicator c_t

    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ using Eq. equation 18

  end for

end for

return π_θ

Table 3: LIBERO success rates with Wilson 95% confidence intervals. Each cell reports success rate followed by the interval in brackets; each suite uses 500 evaluation episodes.

Method	Spatial	Object	Goal	Long
Base Policy	63.6 [59.3, 67.7]	61.6 [57.3, 65.8]	48.6 [44.2, 53.0]	50.8 [46.4, 55.2]
AWR	71.2 [67.1, 75.0]	66.8 [62.6, 70.8]	57.8 [53.4, 62.1]	53.4 [49.0, 57.7]
DPO	66.8 [62.6, 70.8]	64.8 [60.5, 68.9]	51.8 [47.4, 56.1]	51.2 [46.8, 55.6]
CQL	70.4 [66.3, 74.2]	63.0 [58.7, 67.1]	52.6 [48.2, 56.9]	52.0 [47.6, 56.3]
IQL	71.6 [67.5, 75.4]	63.4 [59.1, 67.5]	55.4 [51.0, 59.7]	52.8 [48.4, 57.1]
RedFlow	75.8 [71.9, 79.3]	70.4 [66.3, 74.2]	71.2 [67.1, 75.0]	55.2 [50.8, 59.5]

Table 4: Ablation success rates with Wilson 95% confidence intervals. Each cell reports success rate followed by the interval in brackets; each suite uses 500 evaluation episodes.

Variant	Spatial	Object	Goal
w/o low-quality rollouts	71.4 [67.3, 75.2]	67.4 [63.2, 71.4]	65.2 [60.9, 69.2]
w/o high-quality rollouts	64.4 [60.1, 68.5]	65.8 [61.5, 69.8]	57.0 [52.6, 61.3]
w/o task progress	50.8 [46.4, 55.2]	46.2 [41.9, 50.6]	33.6 [29.6, 37.9]
w/o proprioceptive state	62.4 [58.1, 66.5]	60.2 [55.8, 64.4]	52.0 [47.6, 56.3]
w/o correctability filtering	66.4 [62.1, 70.4]	65.8 [61.5, 69.8]	50.8 [46.4, 55.2]
w/o $\mathcal{L}_{\mathrm{cor}}$	70.4 [66.3, 74.2]	64.4 [60.1, 68.5]	55.6 [51.2, 59.9]
w/o $\mathcal{L}_{\mathrm{sup}}$	70.8 [66.7, 74.6]	66.8 [62.6, 70.8]	68.8 [64.6, 72.7]
w/o both terms	63.8 [59.5, 67.9]	62.8 [58.5, 66.9]	70.4 [66.3, 74.2]
RedFlow	75.8 [71.9, 79.3]	70.4 [66.3, 74.2]	71.2 [67.1, 75.0]

C Simulation Implementation Details

Benchmark and base policy. We evaluate RedFlow on the four LIBERO suites (Liu et al., 2023): Spatial, Object, Goal, and Long, each containing 10 tasks. We adopt π_0 (Black et al., 2024a) initialized from the π_{RL} (Chen et al., 2026a) checkpoint as the base VLA policy. The SFT policy is trained on a pruned expert set $\mathcal{D}_{\mathrm{exp}}$ containing 58 demonstrations for Spatial, Object, and Goal, and 208 demonstrations for Long.

Offline RL data. RL fine-tuning uses

$$\mathcal{D} = \mathcal{D}_{\mathrm{roll}}, \tag{41}$$

where $\mathcal{D}_{\mathrm{roll}}$ contains 1,536 mixed-quality rollout trajectories per suite. The SFT demonstrations described above are used only to initialize the base policy; they are not included in $\mathcal{D}$ and do not contribute positive actions to corrective-target construction. Progress estimates are produced by the pretrained Robo-Dopamine 2.0 GRM, smoothed over temporal chunks, and combined with proprioceptive states for HDBSCAN-based execution-context matching before corrective-target construction. All methods use the same training protocol on each task suite.

Evaluation protocol. We report average success rates over 500 evaluation episodes per suite. The same evaluation protocol is used for all compared methods within each suite.

Evaluation uncertainty. Tables 3 and 4 report Wilson 95% confidence intervals over the 500 evaluation episodes used for each suite and fixed trained policy. These intervals quantify finite-evaluation uncertainty; they do not include variation across independently trained policies or rollout buffers.

Offline baseline implementation. All baselines use the same base policy, offline buffer, action-chunk representation, and Robo-Dopamine 2.0 progress/outcome supervision. The signed chunk-quality score $\hat{A}_t$ in Eq. equation 9 provides the common supervision: AWR uses it to reweight flow-matching regression, DPO uses it to form context-matched preference pairs, and CQL and IQL use $r_t := \hat{A}_t$ as the immediate reward at the action-chunk timescale. Thus, the value-based baselines estimate a discounted return through Bellman updates rather than directly regressing the critic to the instantaneous score.

Let $s_t = (o_t, l)$ and define the per-chunk flow-matching regression error

$$\ell_\theta(s_t, a_t) = \mathbb{E}_{x_1, n}\left[\|v_\theta(x_n, n, o_t, l) - (x_1 - a_t)\|_2^2\right], \tag{42}$$

where $x_n = (1-n)a_t + nx_1$, $x_1 \sim \mathcal{N}(0, \mathbf{I})$, and $n \sim \mathcal{U}(0, 1)$, as in Eq. equation 6. AWR minimizes

$$\mathcal{L}_{\mathrm{AWR}} = \mathbb{E}_{(s_t, a_t)\sim\mathcal{D}}\left[\exp\left(\hat{A}_t / T_{\mathrm{AWR}}\right)\ell_\theta(s_t, a_t)\right], \qquad T_{\mathrm{AWR}} = 3.0, \tag{43}$$

with weight normalization disabled. The score controls the imitation strength of each observed chunk; AWR does not construct corrective targets.

For DPO, each rejected chunk (s_j, a_j^-) is paired with the preferred chunk (s_i, a_i^+) having the smallest Euclidean distance in execution-context space within the same cluster, with $\hat{A}_i > 0$ and $\hat{A}_j < 0$. Each member retains its own observation and instruction; nearest-context retrieval provides the correspondence used for pairing. Following the flow-matching DPO construction used by FlowPRO (Wu et al., 2026), we use the regression error as a surrogate for negative action log-likelihood. Let ℓ_{ref} denote Eq. equation 42 evaluated by the frozen base policy. We define the implicit preference score

$$g_\theta(s, a) = \frac{\beta_{\mathrm{DPO}}}{2}\left[\ell_{\mathrm{ref}}(s, a) - \ell_\theta(s, a)\right], \tag{44}$$

where $\beta_{\mathrm{DPO}} = 0.3$ is the DPO preference scale, and optimize

$$\mathcal{L}_{\mathrm{DPO}} = -\mathbb{E}\left[\log\sigma\big(g_\theta(s_i, a_i^+) - g_\theta(s_j, a_j^-)\big)\right]. \tag{45}$$

The flow times and Gaussian noise used to evaluate the preferred and rejected chunks are sampled independently. This implementation does not require a normalized likelihood from the flow-matching policy. In addition to the contrastive term, the policy receives a behavior-cloning loss on the preferred chunk and a reference-policy KL penalty with coefficient 0.05. It does not use RedFlow's corrective centroid, failure-suppression margin, or target-guided correction loss.

At the chunk level, a transition is represented as (s_t, a_t, s_{t+1}), where $s_t = (o_t, l)$ and $a_t \in \mathbb{R}^{K\times D}$ is executed between s_t and s_{t+1}. Let r_t denote the progress- and outcome-derived reward used by the value-based baselines. Their return and one-step Bellman targets are formed from

$$G_t = \sum_{j=0}^{T-t-1}\gamma^j r_{t+j}, \qquad y_t^{\mathrm{B}} = r_t + \gamma\,\mathcal{V}_{\mathrm{target}}(s_{t+1}), \tag{46}$$

where $\mathcal{V}_{\mathrm{target}}$ denotes the method-specific bootstrapped target. The discount convention is applied at the action-chunk timescale.

Let

$$h_t = H_{\pi_0}^{\mathrm{last}}(o_t, l), \qquad z_t = \phi_a(\mathrm{norm}(a_t)), \qquad Q_\psi(o_t, l, a_t) = g_\psi([h_t; z_t]), \tag{47}$$

where h_t is the final hidden representation of the same π_0 VLM forward pass used by the policy, $a_t \in \mathbb{R}^{K\times D}$ is the complete action chunk, and ϕ_a denotes an MLP projection of the action chunk at the critic input. The concatenated representation $[h_t; z_t]$ is passed through a scalar MLP head g_ψ. Thus the critic is action-conditioned and estimates the expected chunk-level return $Q_\psi(o_t, l, a_t)$ rather than a state-only value from h_t alone. The critic is trained with its Bellman-regression objective using y_t^{B}. For CQL, this objective is augmented with the conservative regularizer

$$\mathcal{L}_{\mathrm{CQL}} = \mathcal{L}_{\mathrm{Bellman}} + \alpha_{\mathrm{CQL}}\left[\log\sum_{a\in\mathcal{A}_t^{\mathrm{sample}}}\exp Q_\psi(s_t, a) - \mathbb{E}_{a\sim\mathcal{D}}Q_\psi(s_t, a)\right], \tag{48}$$

where $\mathcal{A}_t^{\mathrm{sample}}$ contains 10 action chunks in total per state, sampled from the offline data and the policy. We use a single action-conditioned Q-function. The CQL actor is updated to increase the conservative critic value of policy-generated action chunks. If $\tilde{a}_\theta(s_t, x_1)$ denotes the chunk produced by the differentiable flow solver from noise x_1, its actor objective is

$$\mathcal{L}_{\mathrm{actor}}^{\mathrm{CQL}} = -\mathbb{E}_{s_t\sim\mathcal{D},\, x_1\sim\mathcal{N}(0,\mathbf{I})}\left[Q_\psi(s_t, \tilde{a}_\theta(s_t, x_1))\right]. \tag{49}$$

Table 5: Value-based offline baseline hyperparameters. A dash denotes a method-specific parameter that is not used by the corresponding baseline.

Hyperparameter	CQL	IQL
Discount, γ	0.99	0.99
Critic learning rate	3×10^{-4}	3×10^{-4}
Target update, τ_{target}	0.005	0.005
Conservative coefficient, α_{CQL}	1.0	–
Sampled action chunks per state	10	–
Expectile, τ_{exp}	–	0.7
Actor coefficient, β_{IQL}	–	3.0
Maximum actor weight, $w_{\max}$	–	100

Table 6: Training configuration shared by the evaluated offline baselines.

Configuration	Value
Actor update epochs	30
Actor seed	42
Micro/global batch size	128 / 2,048
Actor learning rate	5×10^{-5}
Adam coefficients, (β_1, β_2)	$(0.9, 0.95)$
Adam ϵ	10^{-5}
Weight decay	0.01
Gradient-clipping norm	1.0
Action-chunk length, K	5
Advantage normalization	Disabled
Image augmentation	Disabled
Training / evaluation sampling temperature	1.0 / 0.6
Top-k / top-p sampling	50 / 1.0
Intermediate validation	Disabled

For IQL, let $V_\nu(s_t)$ denote the learned state-value head and define $\delta_t = Q_{\bar{\psi}}(s_t, a_t) - V_\nu(s_t)$, where $Q_{\bar{\psi}}$ is the target critic. The value and critic objectives are

$$\mathcal{L}_V^{\text{IQL}} = \mathbb{E}_{\mathcal{D}}\left[|\tau_{\text{exp}} - \mathbb{1}[\delta_t < 0]|\, \delta_t^2\right], \qquad y_t^{\text{IQL}} = r_t + \gamma V_\nu(s_{t+1}). \tag{50}$$

The critic regresses to y_t^{IQL}, and the flow-matching actor is updated by weighting observed chunks according to their learned return advantage. Its actor weight is

$$w_t^{\text{IQL}} = \min\!\big(\exp\!\big[\beta_{\text{IQL}}\big(Q_\psi(s_t, a_t) - V_\nu(s_t)\big)\big], w_{\max}\big), \tag{51}$$

where $\beta_{\text{IQL}} = 3.0$ and $w_{\max} = 100$. The IQL value head uses an expectile parameter of $\tau_{\text{exp}} = 0.7$.

The same conditional flow-matching actor over $K \times D$ action chunks is used for all methods.

Both value-based baselines use discount $\gamma = 0.99$, critic learning rate 3×10^{-4}, and Polyak target-network updates. Denoting the online critic parameters by ψ and their target-network copy by $\bar{\psi}$, the update is

$$\bar{\psi} \leftarrow (1 - \tau_{\text{target}})\bar{\psi} + \tau_{\text{target}}\psi, \qquad \tau_{\text{target}} = 0.005. \tag{52}$$

For CQL, we use $\alpha_{\text{CQL}} = 1.0$. The baseline-specific values are summarized in Table 5.

Shared baseline training configuration. Except for their method-specific objectives and critic updates, all offline baselines use the same actor initialization, rollout buffer, action-chunk parameterization, and optimization budget. Table 6 summarizes the shared configuration. Intermediate validation is disabled during offline updates, so the evaluation episodes are not used for checkpoint selection within training.

Hyperparameter choices for LIBERO. Table 7 summarizes the offline training configuration used across all four LIBERO suites, and Table 8 reports the HDBSCAN parameters used for

Table 7: Hyperparameters for the single-iteration offline training pipeline of RedFlow on the LIBERO task suites. Spatial, Object, and Goal share the same configuration and SFT initialization. Long uses a longer episode horizon (480 vs. 240), which motivates the remaining changes: λ_{sup} and λ_{cor} are reduced to prevent accumulated suppression/correction signal from dominating the policy update over longer trajectories, while W and b are increased to stabilize progress estimation across more action steps. Bold values indicate settings changed for Long.

Hyperparameter	**Spatial / Object / Goal**	**Long**
Offline training schedule		
Policy update epochs	30	30
Action-level quality estimation		
Progress smoothing window, W	10	**20**
Outcome-bias coefficient, b	0.15	**0.25**
Soft-weight temperature, T_w	3.0	3.0
Attraction coefficient	1	1
Suppression strength, λ_{sup}	0.3	**0.1**
Correction strength, λ_{cor}	0.3	**0.1**
Corrective-target temperature, κ	1.0	1.0
Adaptive margin scale, s_m	1.0	1.0
Adaptive-margin update frequency	Once per optimizer update	
Adaptive-margin aggregation	Mean over the logical batch	
Optimization		
Optimizer	AdamW	
Learning rate	5×10^{-5}	
Adam coefficients, (β_1, β_2)	(0.9, 0.95)	
Weight decay	0.01	
Gradient clipping	1.0	
Micro/global batch size	128 / 2048	
Rollout buffer and action chunks		
Training environments	64	64
Maximum episode length	240	**480**
Action-chunk length, K	5	5
Mixed-quality rollout trajectories	1,536 per suite	
Evaluation episodes	500 per suite	
Initialization and offline data		
Base policy initialization	C_{SOG}	C_{Long}
SFT demonstrations	58	**208**

C_{SOG} = `RLinf-Pi0-SFT-Spatial-Object-Goal`; C_{Long} = `RLinf-SFT-Pi0-LIBERO-Long`. W and b are defined in the chunk-level score $\hat{A}_t$; T_w is the temperature for the soft weight w_t; λ_{sup} controls failure suppression; λ_{cor} controls target-guided correction; κ is the temperature used to form the corrective target $a_t^\star$.

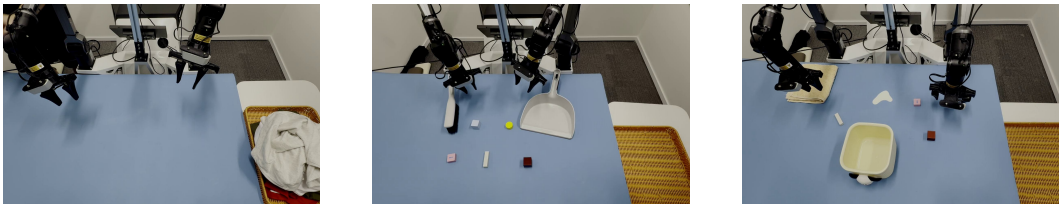
execution-context matching. The Spatial, Object, and Goal suites share an identical configuration, as their episode horizons and task structures are comparable. The Long suite, with twice the maximum episode length (480 vs. 240), requires several adjustments grounded in the role of each hyperparameter.

Action quality estimation (W, b). Longer trajectories produce noisier progress signals at any single step, so we widen the smoothing window W from 10 to 20 and increase the outcome-bias coefficient b from 0.15 to 0.25 to place more weight on episode-level outcomes relative to instantaneous progress.

Suppression strength (λ_{sup}). The choice of λ_{sup} reflects a key property of failed trajectories: not every action chunk in a failed episode is itself harmful, as failures often contain a substantial fraction of

Table 8: HDBSCAN hyperparameters for progress–state execution-context matching in LIBERO. The four LIBERO suites share the same HDBSCAN configuration.

Parameter	Value	Meaning
`min_cluster_size`	15	Minimum HDBSCAN cluster size.
`min_samples`	5	Minimum number of neighbors for a core point; larger values make clustering more conservative.
`progress_weight`, β	5.0	Weight applied to the standardized progress dimension, making task progress more dominant in clustering.
`state_dim`	7	Proprioceptive state dimension used for clustering, excluding the gripper dimension.
Feature preprocessing	`StandardScaler`	Z-score normalization for the state and progress features before clustering.
Centroid softmax temperature	1.0	Temperature of the softmax over the signed chunk-quality scores of positive samples in the same cluster.



(a) Clothes folding (b) Object sweeping (c) Table cleaning

Figure 7: Real-robot experiment setups for bimanual manipulation, tool-mediated interaction, and pick-and-place manipulation.

neutral chunks that simply do not contribute to task progress. An overly large λ_{sup} would indiscriminately suppress these neutral chunks together with the genuinely harmful ones, over-concentrating the policy distribution and destabilizing training. We therefore keep λ_{sup} moderate, and further reduce it from 0.3 to 0.1 on Long, where the suppression signal accumulates over more steps and this effect is more pronounced.

Correction strength (λ_{cor}). The corrective target $a_t^\star$ is not intended to replace the attraction term, which remains the primary driver of policy improvement; rather, λ_{cor} provides a mild bias that shifts the action-sampling distribution toward higher-quality regions. A small value is therefore sufficient and preferred, and we use 0.3 for Spatial/Object/Goal and 0.1 for Long, again to account for the per-step accumulation over longer horizons.

All other hyperparameters, including the optimizer configuration, batch sizes, rollout buffer size, and HDBSCAN corrective-target assignment settings, are shared across all four suites.

D Real-Robot Implementation Details

Hardware and tasks. To evaluate RedFlow beyond simulation, we conduct real-world experiments on a dual-arm Agilex Cobot Magic robot equipped with three cameras, including one front-facing camera and two wrist-mounted cameras for visual input. We consider three tasks of increasing manipulation diversity: clothes folding, object sweeping, and table cleaning, covering dexterous bimanual manipulation, tool-mediated interaction, and pick-and-place manipulation, respectively.

Base policy and offline RL data. We initialize from the official π_0 base checkpoint provided by OpenPi (Black et al., 2024a). For each real-world task, we obtain the base policy by fine-tuning on task-specific expert demonstrations for 50,000 steps. The expert demonstration counts are 600 for clothes folding, 200 for object sweeping, and 100 for table cleaning. For offline RL training, we

Table 9: Real-robot task instructions used for policy conditioning.

Task	Instruction
Clothes Folding	`fold the cloth`
Object Sweeping	`Use the brush to sweep 5 small blocks into the dustpan one by one, then empty them into the basket beside the table and push down the brush on the table.`
Table Cleaning	`Pick up 3 blocks and put them in the yellow box one by one, then use the cleaning cloth to wipe up the milk spill.`

collect 200, 100, and 100 rollouts from the initial policy for clothes folding, object sweeping, and table cleaning, respectively, yielding mixed-quality rollout buffers.

Execution-context matching. For real-robot experiments, progress predictions from the Robo-Dopamine 2.0 GRM are computed with `frame_interval=10`. Execution-context matching uses HDBSCAN with `min_cluster_size=50`, `min_samples=10`, and progress weight $\beta = 5.0$.

Evaluation protocol. We evaluate each task over 100 rollouts, which provides substantially more reliable success-rate estimates than the 10–20 rollouts common in prior real-world VLA evaluations. Real-world episodes are not constrained by a fixed step limit; instead, each rollout terminates upon one of two conditions: (i) the task-specific success criterion is satisfied, or (ii) the policy enters an unrecoverable failure state in which a manipulated object is dropped outside the workspace or in a configuration from which the policy cannot recover. Between rollouts, the scene is manually reset to a randomized initial configuration drawn from a task-specific distribution over object positions and orientations.

Success criteria are defined per task as follows. **Clothes folding:** the robot folds the cloth into the target folded configuration without the garment falling off the table. **Object sweeping:** the robot uses the brush to sweep all 5 small blocks into the dustpan one by one, empties them into the basket beside the table, and pushes down the brush on the table. **Table cleaning:** the robot picks up all 3 blocks and places them into the yellow box one by one, then uses the cleaning cloth to wipe up the milk spill. A rollout that does not satisfy the corresponding success criterion before terminating is counted as a failure.

Real-robot evaluation uncertainty. Table 11 reports exact success counts and Wilson 95% confidence intervals over the 100 trials per task. As in simulation, these intervals quantify evaluation uncertainty for each fixed trained policy and do not capture training-seed variation.

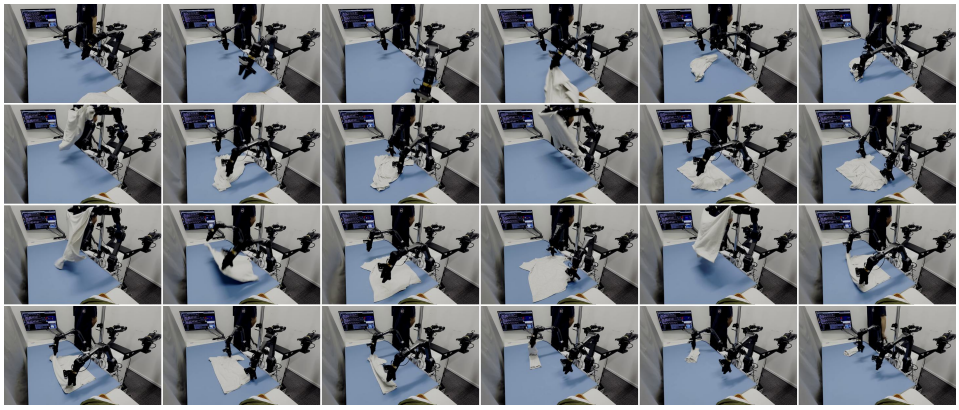


Figure 8: Successful trajectory of clothes folding.

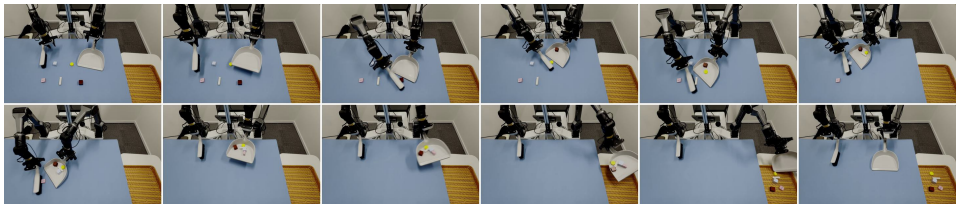


Figure 9: Successful trajectory of object sweeping.

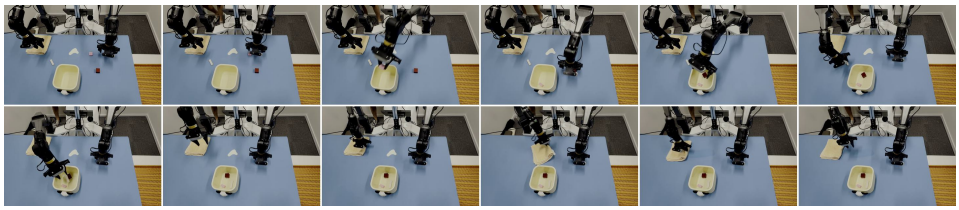


Figure 10: Successful trajectory of table cleaning.

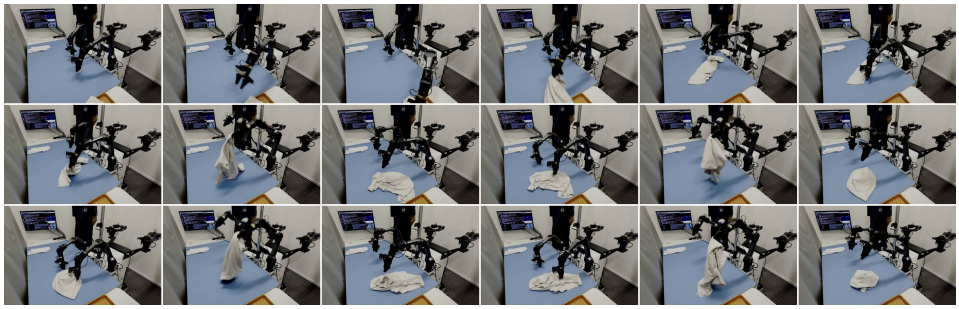


Figure 11: Failed trajectory of clothes folding.

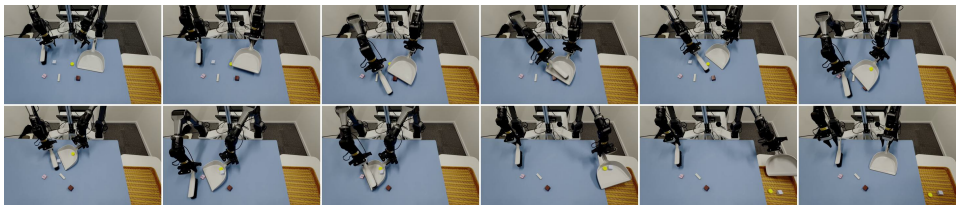


Figure 12: Failed trajectory of object sweeping.



Figure 13: Failed trajectory of table cleaning.

Table 10: Hyperparameters and setup for real-world experiments. All three tasks share the same π_0 LoRA backbone, Quality-Guided Action Redirection formulation, optimizer, and distributed training configuration; task-specific items are listed per task.

Hyperparameter	**Clothes Folding**	**Object Sweeping**	**Table Cleaning**
Hardware and control			
Robot platform	Agilex Cobot Magic (dual-arm)		
Control frequency	100 Hz		
Cameras	`cam_high`, `cam_left_wrist`, `cam_right_wrist`		
Camera model	Intel RealSense D435		
Camera frequency	30 Hz		
Action dimension, D	32		
Action-chunk length, K	50		
Backbone and initialization			
Vision-language backbone	`gemma_2b_lora`		
Action expert backbone	`gemma_300m_lora`		
Numerical precision	`bfloat16`		
Maximum language-token length	64		
Base initialization	`gs://openpi-assets/checkpoints/pi0_base/params`		
Task-specific initialization	`checkpoints/real_task/{task}/params`		
Task fine-tuning steps	50,000		
Normalization-statistics asset	`folding_cloth`	`aloha_sweeping`	`aloha_cleaning`
Progress estimation and corrective targeting			
GRM prediction frame interval	10		
`min_cluster_size`	50		
`min_samples`	10		
Progress weight, β	5.0		
Quality-Guided Action Redirection			
Soft-weight temperature, T_w	3.0		
λ_{sup}	0.3		
Adaptive margin scale, s_m	1.0		
λ_{cor}	0.3		
Adaptive-margin update frequency	Once per optimizer update		
Adaptive-margin aggregation	Mean over the logical batch		
Offline adaptation and optimization			
Policy update epochs	30		
Batch size	32		
Optimizer	AdamW with cosine warmup		
Peak learning rate	2.0×10^{-5}		
Final learning rate	2.5×10^{-6}		
Warmup / cosine decay steps	1,000 / 30,000		
Adam coefficients, (β_1, β_2)	(0.9, 0.95)		
Weight decay	1.0×10^{-10}		
Gradient-clipping norm	1.0		
FSDP devices	8		
Data and evaluation			
Expert demonstrations	600	200	100
Offline RL rollouts	200	100	100
Evaluation rollouts	100	100	100

Table 11: Real-robot success counts with Wilson 95% confidence intervals. Each cell reports successes out of 100 trials followed by the success-rate interval in brackets.

Method	Clothes folding	Object sweeping	Table cleaning
Base Policy	36/100 [27.3, 45.8]	63/100 [53.2, 71.8]	71/100 [61.5, 79.0]
AWR	48/100 [38.5, 57.7]	66/100 [56.3, 74.5]	78/100 [68.9, 85.0]
DPO	41/100 [31.9, 50.8]	69/100 [59.4, 77.2]	76/100 [66.8, 83.3]
RedFlow	67/100 [57.3, 75.4]	73/100 [63.6, 80.7]	84/100 [75.6, 89.9]

E CHUNK-QUALITY AND CORRECTIVE-TARGET DIAGNOSTICS

Utility of the signed chunk-quality construction. We first isolate the two components of Eq. equation 9. Let $\Delta\bar{p}_t = \bar{p}_{\min(t+W,T-1)} - \bar{p}_{\max(t-W,0)}$. The progress-only variant uses $\hat{A}_t = \Delta\bar{p}_t$, while the outcome-only variant uses $\hat{A}_t = b(2\mathbb{1}[y_\tau = 1] - 1)$. The full variant combines both terms. As a sanity check, the shuffled-outcome variant retains the progress term but randomly permutes trajectory outcomes across the buffer while preserving their marginal frequency. All other data, optimization, and evaluation settings are held fixed.

Table 12: Utility ablation of the signed chunk-quality construction on LIBERO. Coverage is the percentage of negative chunks assigned a corrective target. The average is computed over all four suites.

Score construction	Spatial	Object	Goal	Long	Avg.	Target coverage (%)
Progress only	71.4	66.2	64.4	46.8	62.2	9.6
Outcome only	62.6	60.8	48.0	44.8	54.1	21.7
Progress + outcome	**75.8**	**70.4**	**71.2**	**55.2**	**68.2**	11.6
Shuffled outcome	48.2	41.4	36.2	26.8	38.2	20.4

Progress-only supervision substantially outperforms outcome-only supervision, showing that local progress change provides useful temporal information beyond the final trajectory label. Combining both terms performs best, improving the four-suite average over progress only and outcome only by 6.0 and 14.1 points, respectively. Shuffling the trajectory outcomes reduces performance to 38.2% despite increasing target coverage to 20.4%, indicating that improvement does not follow from assigning more corrective targets alone; consistency between progress and trajectory outcome matters. This ablation evaluates the downstream utility of the score construction. It does not treat $\hat{A}_t$ as ground-truth causal action quality.

Table 13: Diagnostics of negative-chunk usage on LIBERO. All negative chunks are used for suppression; only negative chunks whose progress–state cluster contains positive chunks also receive target-guided correction. We report the fractions of negative chunks optimized with suppression only and with suppression plus correction, as well as the mean signed chunk-quality score of the positive chunks used to construct each target, $\hat{A}^\star = \sum_{i \in \mathcal{C}_c^+} \alpha_i \hat{A}_i$. The average score pools target-bearing negative chunks across suites rather than averaging the suite-level scores.

Suite	Suppression only (%)	Supp. + Corr. (%)	Mean $\hat{A}^\star$
Spatial	86.3	13.7	$+0.216$
Object	89.5	10.5	$+0.274$
Goal	89.3	10.7	$+0.256$
Avg	88.4	11.6	$+0.243$

Mechanism-isolation summary. The following contrasts are derived from the ablations in Table 2. They use the same reported LIBERO evaluation values and make explicit the performance contribution of each redirection component. The full method is compared against variants that remove only the indicated component; therefore, these contrasts should be interpreted as controlled ablation effects rather than as additional training runs.

The comparison with the variant that removes both negative-chunk terms is particularly important: this variant retains the same quality-weighted attraction mechanism, so the 6.8-point difference measures the additional benefit of failure suppression and target-guided redirection beyond quality reweighting alone. The 9.0-point difference obtained by removing only $\mathcal{L}_{\text{cor}}$ further shows that repulsion alone is insufficient: the locally supported corrective target provides the direction needed to make failure suppression useful. Finally, removing correctability filtering causes an 11.5-point drop, supporting the use of positive support as a guard against assigning unsupported corrective references. Since only 11.6% of negative chunks receive correction on average, RedFlow applies this constructive signal selectively rather than treating every negative chunk as correctable.

Table 14: Mechanism-isolation summary derived from the LIBERO ablations in Table 2. The reference is the full objective at 72.5% average success. All values are averaged over Spatial, Object, and Goal.

Variant	Avg. success (%)	Change	Interpretation
Full RedFlow	72.5	–	Quality weighting, suppression, and correction.
w/o $\mathcal{L}_{\text{cor}}$	63.5	-9.0	Suppression without target-guided correction.
w/o $\mathcal{L}_{\text{sup}}$	68.8	-3.7	Target attraction without explicit failure suppression.
w/o both terms	65.7	-6.8	Quality-weighted flow matching without redirection.
w/o correctability filtering	61.0	-11.5	Indiscriminate target assignment without positive-support filtering.

Target-quality diagnostics. To test whether the corrective targets remain close to successful behavior, we measure action-space statistics for negative chunks that receive a target. All action chunks are flattened and normalized using the policy action statistics; distances are divided by $\sqrt{KD}$. For a matched cluster $\mathcal{C}_c^+$, let $a_c^\star = \sum_{i\in\mathcal{C}_c^+} \alpha_i a_i$ denote its weighted positive-action centroid. We compute positive-action dispersion as

$$D_c = \sqrt{\frac{1}{KD} \sum_{i\in\mathcal{C}_c^+} \alpha_i \|a_i - a_c^\star\|_2^2}, \tag{53}$$

the centroid-to-nearest-positive distance as

$$d_{\text{pos}}(c) = \min_{i\in\mathcal{C}_c^+} \frac{\|a_c^\star - a_i\|_2}{\sqrt{KD}}, \tag{54}$$

and the negative-to-target distance as

$$d_{\text{neg}}(t) = \frac{\|a_t - a_c^\star\|_2}{\sqrt{KD}}. \tag{55}$$

All three statistics are aggregated over negative chunks that receive a corrective target.

Table 15: Target-quality diagnostics for negative chunks that receive corrective targets. Action chunks are normalized before computing distances, and each distance is divided by $\sqrt{KD}$.

Suite	Coverage (%)	Positive dispersion	Centroid–nearest positive	Negative–target
Spatial	13.7	0.074	0.042	0.317
Object	10.5	0.085	0.067	0.411
Goal	10.7	0.102	0.071	0.528
Average	11.6	0.087	0.060	0.419

Across the three suites, positive actions within matched clusters have an average dispersion of 0.087, while the weighted centroid is on average only 0.060 from its nearest positive action. In contrast, the original negative action is 0.419 from the corrective target on average. The substantially smaller centroid-to-positive distance indicates that the target remains close to observed successful behavior, whereas the larger negative-to-target distance shows that it provides a nontrivial redirection away from the failure action. These diagnostics support interpreting the centroid as a locally supported corrective reference rather than an arbitrary average of unrelated actions.

Illustrative endpoint-redirection case study. For the negative chunk illustrated in Figure 3, we additionally compare the reconstructed clean endpoint produced by the base policy and by RedFlow under the same endpoint-evaluation input. Using the normalized RMS Euclidean metric from

Eqs. equation 54– equation 55, we define

$$d_{\text{fail}} = \frac{\|\hat{x}_0 - a_t^-\|_2}{\sqrt{KD}}, \qquad d_{\text{target}} = \frac{\|\hat{x}_0 - a_t^\star\|_2}{\sqrt{KD}}. \tag{56}$$

Table 16 reports the resulting distances.

Table 16: Endpoint redirection for the illustrative corrective-matching case in Figure 3. Smaller distance indicates that the predicted endpoint is closer to the corresponding action chunk.

Policy	d_{fail}	d_{target}	$d_{\text{fail}} - d_{\text{target}}$
Base policy	0.042	0.079	-0.037
RedFlow	0.065	0.032	$+0.033$

Relative to the base policy, RedFlow reduces the distance to the corrective target by 59.5% and increases the distance from the observed failure action by 54.8%. Accordingly, the relative-distance gap $d_{\text{fail}} - d_{\text{target}}$ changes sign: the base endpoint is closer to the failure action, whereas the RedFlow endpoint is closer to the matched corrective target. This single case is an illustration of the learned redirection mechanism rather than a population-level guarantee.

Table 17: Comparison of corrective-target constructions on LIBERO. The average is computed over Spatial, Object, and Goal. The final row is the existing ablation without target-guided correction and is included as a reference.

Target construction	Spatial	Object	Goal	Average
Corrective target	75.8	70.4	71.2	72.5
Nearest positive action	66.4	63.2	57.2	62.3
Positive-action medoid	70.8	68.2	58.6	65.9
No target-guided correction	70.4	64.4	55.6	63.5

The weighted centroid achieves 72.5% average success, exceeding the nearest-positive and medoid constructions by 10.2 and 6.6 percentage points, respectively. Removing target-guided correction reduces the average to 63.5%, consistent with the objective ablation reported in Table 2. Together, these results indicate that aggregating multiple supported positive actions is more effective than selecting a single positive action in this setting, while explicit target-guided correction contributes beyond the other loss terms.

F LIMITATIONS

RedFlow is designed for offline post-training with a fixed rollout buffer, so its behavior naturally reflects the coverage of the collected experience and the quality of the progress and execution-context representations used for matching. Because the execution context omits full visual and contact state, co-clustered chunks provide local support rather than verified physical-state equivalence. The present design favors a compact context representation that is effective and practical for the evaluated tasks; richer visual, geometric, or temporal features may further improve target selection in more diverse settings. Confidence-aware matching and adaptive target construction are also promising extensions when positive support is sparse or ambiguous. Finally, our evaluation covers LIBERO and three real-robot manipulation tasks. Extending the study to additional embodiments, environments, and longer-horizon tasks would provide a broader characterization of the method's generality.