arXiv:2607.27782v1 [cs.RO] 30 Jul 2026

RedFlow: Redirect Failure into Action-Level Corrections for Flow-matching VLA Policy

Zhengyang Yan[†] **Junhao Li**[†] **Fangqi Zhu**[†] **Zijun Wang** **Quanxin Shou**
Yikun Miao **Zicong Hong** **Xiaoyi Pang** **Song Guo**[*]

The Hong Kong University of Science and Technology
zhengyang.yan@connect.ust.hk, fzhuah@connect.ust.hk
[†]Equal contribution. [*]Corresponding author.

Abstract

Flow-matching Vision-Language-Action (VLA) policies show great potential for robotic manipulation but frequently suffer from compounding errors caused by distribution shifts during deployment. While offline Reinforcement Learning (RL) provides a practical way to mitigate this issue by learning from deployment rollouts, existing methods either ignore failure data or use it only at a coarse trajectory level, leading to low learning efficiency and persistent errors. To solve this issue, we propose **RedFlow**, a fine-grained offline RL framework that **Red**irects failure experiences into high-fidelity action-level correction signals for **Flow**-matching VLA policies. RedFlow has two key components: a *Context-Aware Corrective Matching* mechanism that identifies failure-inducing actions and retrieves successful alternatives from similar contexts as local corrective targets, and an *Adaptive Redirection Objective* that modulates the training signal at three complementary levels—reinforcing successful actions, suppressing undesirable ones, and redirecting recoverable failures toward corrective targets. By turning both successes and failures into dense, structured supervision, RedFlow enables the policy to learn robust recovery behaviors from mixed-quality data. On the LIBERO benchmark and three real-robot manipulation tasks, RedFlow consistently outperforms state-of-the-art offline RL baselines, lifting the real-world success rate from 56.7% to 74.7%. The trained policy further exhibits emergent recovery behaviors that the base policy fails to produce—for instance, using the opposite arm to retrieve an out-of-reach object before retrying the task. Notably, RedFlow matches the performance of strong on-policy baselines (PPO, GRPO, DDPO) while using approximately an order of magnitude fewer training samples, establishing structured failure reuse as a sample-efficient direction for the post-training of generalist VLA policies.

1 Introduction

Vision-Language-Action (VLA) policies built upon flow matching [1–3] have recently emerged as a powerful paradigm for generalist robotic manipulation. Flow matching learns a velocity field that maps noise to actions, enabling policies to model multimodal action distributions and capture the diverse strategies inherent in real-world tasks. However, since these policies are normally trained via imitation learning (IL) on human demonstrations, they inherit a fundamental limitation of behavior cloning: compounding errors under distribution shift. During real-world deployment, when the robot encounters states that deviate from the training distribution, small prediction errors accumulate over successive steps and eventually drive the policy into unrecoverable failures.

Reinforcement Learning (RL) [4] enables robots to bridge the gap between training and deployment via direct learning from environment interactions. While online methods [5–7] offer continuous im-

Preprint.

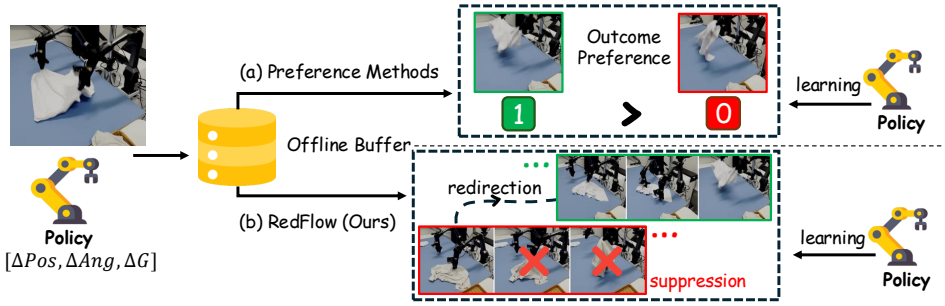


Figure 1: **RedFlow combines two properties that prior methods lack: learning from failures and action-level guidance.** (a) Preference methods learn from failures but only at the trajectory level, signaling what to avoid without action-level corrections. (b) RedFlow identifies failure-inducing actions within failed rollouts and redirects them toward corrective targets derived from successful experiences, providing dense action-level supervision.

provement by continually collecting and learning from fresh on-policy rollouts, their high interaction costs on real robots limit scalability. Offline RL provides a sample-efficient alternative by learning from pre-collected deployment rollouts. However, existing offline approaches each leave one of two key properties unmet (Fig. 1). Classical offline RL methods [8, 9] primarily reweight or imitate successful behaviors, *failing to learn from failures* and discarding the rich diagnostic information they contain. Preference-based methods [10, 11] do learn from failures, but only through coarse trajectory-level comparisons that *lack action-level guidance*: they signal what to avoid without specifying how the policy should improve. Human-in-the-loop interventions [12–15] can offer both properties, but require expert effort that does not scale. Consequently, a significant gap remains: current frameworks lack a scalable mechanism that can extract high-fidelity action-level corrections from pre-collected failure experiences without human intervention. This motivates our central question: **How can we precisely redirect failure experiences into action-level corrections that go beyond mere repulsion from bad actions and provide explicit supervisory targets for policy improvement?**

We identify two fundamental challenges. The first one lies in the *granularity mismatch* between failure labels and the required policy updates. A trajectory-level failure label obscures the sparse, action-level errors that actually cause the collapse, as many intermediate actions within a failed rollout remain perfectly reasonable. Consequently, it is inherently difficult to isolate specific failure-inducing steps and extract their exact corrective targets without dense human supervision. The second challenge is how to rigorously and precisely integrate these corrective signals into the learning process. Conventional flow-matching policies treat all provided actions as valid samples to be imitated. This "uniform imitation" is problematic when dealing with mixed-quality data: the policy may inadvertently learn sub-optimal behaviors or suffer from over-correction. Avoiding such cases when leveraging corrective signals is challenging.

To address these challenges, we propose **RedFlow**, a fine-grained offline RL framework that **Red**irects failures into action-level correction signals for **Flow**-matching VLA policies. RedFlow consists of two components, each targeting one of the challenges above. First, to resolve the granularity mismatch, we propose a *Context-Aware Corrective Matching* mechanism. It defines an *execution context* using the robot's proprioceptive state and task progress signal, identifies candidate failure-inducing action points within it, and retrieves alternatives from successful experiences in similar execution contexts. These alternatives are treated as local *corrective targets*: they are not assumed to be one-to-one replacement actions, but provide constructive directions for redistributing probability mass away from failure-prone regions. Second, to integrate these corrective signals precisely, we introduce an *Adaptive Redirection Objective* that modulates the training signal at three complementary levels based on the quality of the matched guidance: it reinforces high-quality successful actions, suppresses undesirable ones, and redirects recoverable failures toward corrective targets in the policy's velocity field. This adaptive objective avoids the uniform imitation pitfall and aligns the corrective signals with the flow-matching velocity-field parameterization. By turning both successes and failures into dense, structured supervision, RedFlow improves sample efficiency without requiring additional human demonstrations or online interactions.

Our contributions are summarized as follows: **(i)** We propose RedFlow, a fine-grained offline RL framework that systematically transforms failure trajectories into action-level corrective learning signals for flow-matching VLA policies, enabling the policy to learn robust recovery behaviors without requiring additional human demonstrations or online interactions. **(ii)** We introduce a dual-component precision redirection approach: a context-aware matching procedure that identifies candidate failure points and derives high-fidelity corrective targets from successful experiences, and an adaptive redirection objective that aligns these corrective signals with the policy's velocity-field parameterization. **(iii)** We validate RedFlow on the LIBERO benchmark and three real-robot manipulation tasks, where it surpasses state-of-the-art offline RL baselines (AWR, DPO) by clear margins and reaches comparable performance to strong online RL baselines (PPO, GRPO, DDPO) at a fraction of the rollout cost.

2 Related Work

Flow-matching VLA Policy Improvement. Vision-language-action policies with diffusion or flow-matching action heads have become a powerful paradigm for generalist robotic manipulation, as they can model multimodal continuous action distributions and inherit strong priors from large-scale vision-language backbones [1–3, 16–19]. However, these policies are commonly trained by imitation learning, which treats demonstrations as uniformly desirable and provides limited mechanisms for correcting deployment failures under distribution shift. RL can further improve VLA policies by optimizing behavior on encountered states, but online approaches require repeated environment interaction and are costly for real-world robots [5–7, 20, 21]. Offline methods [8, 22–24] avoids additional interaction by learning from fixed datasets, yet standard objectives often emphasize successful behaviors, neglecting failure behaviors, or suppress undesirable actions without specifying "what the policy should do". In contrast, RedFlow targets flow-matching VLA policy post-training in a strictly offline setting, where both successful and failed rollouts are exploited for policy improvement.

Failure-Aware Learning from Process Feedback. Failed trajectories provide valuable information about policy errors, but trajectory-level failure labels are too coarse to identify which intermediate actions caused the failure. To extract finer-grained signals from execution, prior work has explored various forms of reward modeling for robotic policy learning [25–27]. These methods can assess whether a trajectory or state is moving toward success, but they remain primarily evaluative and do not specify how a continuous action should be corrected. Preference-based objectives also make use of mixed-quality data by favoring successful trajectories over failed ones [8, 10, 11, 28]. However, these methods typically guide learning through rankings, weights, or likelihood modulation, rather than producing explicit action-level corrections. Human-in-the-loop correction can provide direct action-level supervision, but requires expert intervention [12, 13, 15], limiting its scalability. RedFlow addresses this gap by jointly using progress and proprioceptive state information to transform offline failure data into action-level corrective supervision. This enables recoverable failures to be constructively redirected without additional rollouts or human corrections.

3 Methodology

3.1 Preliminaries

Problem Setup We formulate language-conditioned robotic manipulation as an MDP over observation space $\mathcal{O}$ and 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 under the task return. For a language-conditioned manipulation task, the standard RL objective is

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

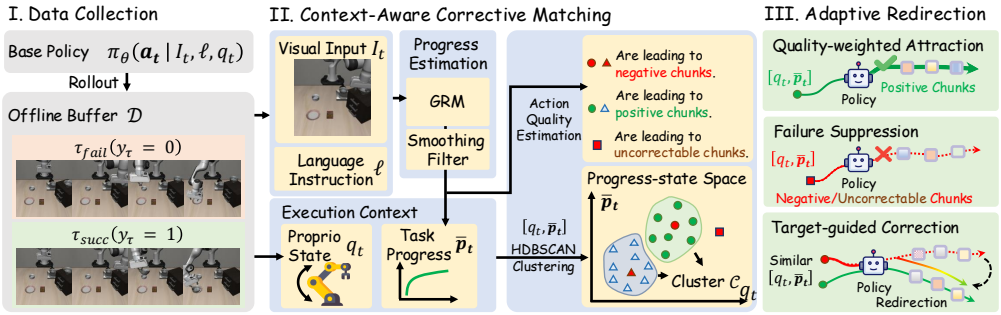


Figure 2: **RedFlow pipeline.** RedFlow consists of three stages: **(I) Data Collection**, where successful and failed rollouts are stored in an offline buffer; **(II) Context-Aware Corrective Matching**, which leverages a pre-trained GRM to estimate task progress, combines progress estimates with proprioceptive states to define execution contexts, clusters similar contexts with HDBSCAN, and derives corrective targets from positive actions observed in matched contexts; and **(III) Adaptive Redirection Objective**, where asymmetric flow matching reinforces high-quality chunks, suppresses negative chunks, and redirects correctable failure-inducing chunks.

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 advantage

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

Actions with positive advantage should be made more likely, whereas actions with negative advantage should be down-weighted or avoided, since such updates locally improve the expected return in Eq. (1).

Flow Matching VLA Policy We instantiate π_θ as a conditional flow-matching model [29]. Starting from noise $x_1 \sim \mathcal{N}(0, \mathbf{I})$, the policy integrates a learned velocity field $v_\theta(x_n, n, o_t, l)$ along the flow timestep $n \in [0, 1]$ to produce the action chunk $a_t = x_0$, ie, by solving the ODE $\mathrm{d}x_n/\mathrm{d}n = v_\theta(x_n, n, o_t, l)$ from $n = 1$ to $n = 0$. The velocity field is pretrained on a dataset of expert demonstrations $\mathcal{D}_{\text{exp}}$ using the standard flow-matching objective

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{n,\, x_0 \sim \mathcal{D}_{\text{exp}}} \left[\| v_\theta(x_n, n, o_t, l) - u_n \|^2 \right], \tag{3}$$

where u_n is the conditional vector field.

3.2 Context-Aware Corrective Matching

Although Eq. (1) defines the desired return-maximization objective, the fixed buffer $\mathcal{D}$ does not provide the true chunk-level advantage in Eq. (2). The available trajectory-level outcomes are too coarse to identify which individual chunks should be reinforced or suppressed. We therefore estimate a proxy advantage for each action chunk using learned task-progress estimates and trajectory outcomes.

Action-level advantage estimation. To obtain an offline surrogate for the chunk-level advantage, we convert learned task-progress estimates and trajectory outcomes into a signed score. We use a pretrained General Reward Model (GRM) [27], $R(o_t, l) \in [0, 1]$, as the source of task-progress estimates from the observation and instruction. Since raw GRM scores may fluctuate across nearby chunks, we smooth the progress sequence with a box filter of half-window size $W > 0$:

$$\bar{p}_t = \frac{1}{2W + 1} \sum_{j=t-W}^{t+W} R(o_j, l), \tag{4}$$

with one-sided averaging at the boundaries. We then convert the smoothed progress sequence into a signed action-level score by combining local progress change with a trajectory-level outcome bias:

$$\hat{A}_t = \bar{p}_{t+W} - \bar{p}_{t-W} + b \cdot \big(2\, \mathbb{1}[y_\tau{=}1] - 1\big), \tag{5}$$

where $b > 0$ is a fixed coefficient and y_τ is the outcome label of the trajectory containing a_t. The local progress-change term captures whether the trajectory advances around chunk a_t, while the outcome-bias term injects coarse success or failure information when local progress alone is ambiguous. We clip the indices $t - W$ and $t + W$ to valid trajectory boundaries. We use the sign of $\hat{A}_t$ as the action-level label: chunks with $\hat{A}_t > 0$ are labeled positive, chunks with $\hat{A}_t < 0$ are labeled negative, and the zero case is handled through the soft weight in Eq. (9) rather than used for corrective-target assignment.

Context clustering for corrective targets. To contextualize these action labels, we define a chunk's execution context via the task instruction, normalized proprioceptive state, and smoothed progress estimate, and operationalize it for clustering through the progress–state feature below. The task instruction is fixed within each task-specific clustering problem, while the progress–state feature captures local execution stage and robot configuration. Specifically, we define the progress–state space using the proprioceptive state q_t and the smoothed progress estimate $\bar{p}_t$. Our key observation is that chunks close in this space often correspond to the same underlying subtask, even when they come from trajectories with different final outcomes.

The proprioceptive state captures the robot configuration, while the GRM-predicted progress implicitly incorporates task-relevant visual information. This design avoids direct clustering in high-dimensional visual space.

We instantiate this space with the feature

$$f_t = [\, \tilde{q}_t;\ \beta\, \bar{p}_t \,], \tag{6}$$

where $\tilde{q}_t$ denotes normalized proprioception and $\beta > 0$ balances the relative scales of task progress and proprioceptive state. For each task, we cluster all action chunks in $\mathcal{D}$ using HDBSCAN [30] over $\{f_t\}$, yielding clusters $\{\mathcal{C}_c\}$.

For a cluster $\mathcal{C}_c$, let $\mathcal{C}_c^+ = \{\, i \in \mathcal{C}_c : \hat{A}_i > 0 \,\}$ denote its positive subset. For each negative chunk a_t with $\hat{A}_t < 0$ assigned to $\mathcal{C}_c$, if $\mathcal{C}_c^+$ is non-empty, we construct a corrective target as a quality-weighted action centroid of the positive chunks in the same cluster:

$$\alpha_i = \frac{\exp(\hat{A}_i/\kappa)}{\sum_{j \in \mathcal{C}_c^+} \exp(\hat{A}_j/\kappa)}, \qquad a_t^\star = \sum_{i \in \mathcal{C}_c^+} \alpha_i a_i, \tag{7}$$

where $\kappa > 0$ controls the concentration of the centroid toward high-advantage chunks. A smaller κ makes the corrective target closer to the highest-advantage positive actions, while a larger κ yields a more uniform cluster centroid. Crucially, a_t^* acts not as a one-to-one counterfactual replacement, but as an empirical positive barycenter defining a local transport direction to redistribute probability mass away from failure modes. A negative chunk is designated *uncorrectable* if it is marked as an outlier by HDBSCAN or if $\mathcal{C}_c^+ = \emptyset$. Such chunks lack corrective targets and are only suppressed during training, as described in Section 3.3.

3.3 Adaptive Redirection Objective

Applying the standard flow-matching objective (Eq. 3) uniformly clones both desirable and failure-prone behaviors. We instead introduce an asymmetric redirection objective.

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

$$\phi(h) = \lambda_{\mathrm{cor}} \|h - b\|^2 + \lambda_{\sup} [m - \|h - a^-\|^2]_+ \tag{8}$$

where $[r]_+ = \max(r, 0)$. The minimizer of ϕ is the closest endpoint to the corrective target that remains outside the finite margin region around a^-. Equivalently, the solution is the projection of b onto the complement of the obstacle ball $\{h : \|h - a^-\|^2 < m\}$. The formal statement and proof are given in Appendix A, Theorem 6.

As formalized in Appendix A, this objective induces a local Wasserstein push-pull transport: $\mathcal{L}_{att}$ anchors to positive endpoints, $\mathcal{L}_{sup}$ bounds finite-range exclusion, and $\mathcal{L}_{cor}$ directs corrective transport.

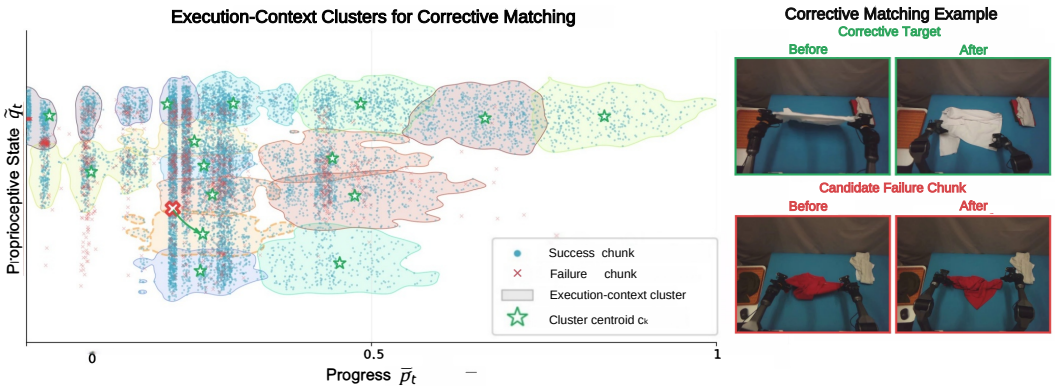


Figure 3: **Context-aware corrective matching in execution-context space.** RedFlow represents each action chunk by its execution context $(\tilde{q}_t,\ \bar{p}_t)$, combining the robot's proprioceptive state and task progress. Action chunks with similar contexts are grouped into execution-context clusters. Given a candidate failure chunk (red cross), RedFlow retrieves matched successful chunks from the corresponding cluster and uses them as corrective targets. The real-robot example illustrates the rationale: although the successful and failure chunks begin from similar contexts, only the matched successful chunk flattens the garment, while the failure chunk leaves it crumpled.

Quality-weighted attraction. We first convert the estimated action-level score $\hat{A}_t$ into a soft weight

$$w_t = \sigma(\hat{A}_t / T_w) \in (0, 1), \tag{9}$$

where $T_w > 0$ is a temperature. Confidently positive chunks receive weights close to one, while confidently negative chunks receive weights close to zero. We then use w_t to modulate the standard flow-matching loss:

$$\mathcal{L}_{\text{att}} = w_t \cdot \|v_\theta(x_n, n, o_t, l) - u_n\|^2. \tag{10}$$

This term attracts the policy toward high-quality chunks while still providing a weak data-support signal for low-quality chunks.

Failure suppression. For chunks estimated to be low-quality, attraction alone is insufficient: the policy should also reduce the tendency to reproduce failure-inducing actions. We therefore introduce a repulsive hinge loss on the reconstruction error

$$e_t = \|\hat{x}_0 - a_t\|^2, \qquad \hat{x}_0 = x_n - n \cdot v_\theta(x_n, n, o_t, l), \tag{11}$$

where $\hat{x}_0$ denotes the predicted clean action under the linear flow-matching interpolation. The suppression loss is

$$\mathcal{L}_{\text{sup}} = \lambda_{\text{sup}}(1 - w_t)\max(0, m - e_t), \tag{12}$$

where $\lambda_{\text{sup}} > 0$ controls the suppression strength and m is an adaptive margin, implemented as a stop-gradient running average of the reconstruction error. Minimizing this hinge loss increases the distance between the predicted action and the negative chunk only when they are closer than m. Once the prediction is sufficiently far from the negative action, the repulsive penalty becomes inactive.

Target-guided correction. Suppression prevents the policy from reproducing a negative chunk, but it does not specify where the local action distribution should move. For correctable failure chunks, we therefore add an attractive correction term toward the cluster-derived target $a_t^\star$:

$$\mathcal{L}_{\text{cor}} = c_t \cdot \lambda_{\text{cor}}(1 - w_t)\|\hat{x}_0 - a_t^\star\|^2, \tag{13}$$

where $\lambda_{\text{cor}} > 0$ controls the correction strength and $c_t \in \{0, 1\}$ indicates whether chunk a_t is a correctable failure chunk with an assigned target. This term should not be interpreted as cloning $a_t^\star$ as the exact corrective action for the state that produced a_t. Rather, it provides a bounded endpoint-level bias that moves probability mass toward locally supported positive regions while the attraction term keeps training anchored to observed behavior data.

Total objective. The final training objective is

$$\mathcal{L} = \mathbb{E}_{(o_t, a_t, l) \sim \mathcal{D},\, n} \left[\mathcal{L}_{\text{att}} + \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{cor}} \right]. \tag{14}$$

The three terms play complementary roles: quality-weighted attraction reinforces positive behavior, suppression discourages failure-inducing actions, and target-guided correction redirects correctable failures toward positive alternatives. 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. We then derive chunk-level advantages and corrective targets using the procedure in Section 3.2. Finally, we optimize the policy on the frozen buffer using the adaptive redirection objective.

4 Experiment

We conduct experiments to evaluate the effectiveness of RedFlow, an offline policy-improvement framework that redirects failure trajectories into action-level corrective supervision while also exploiting successful trajectories. Our experiments are designed to answer the following questions: (1) How does RedFlow compare to existing offline RL baselines on mixed-quality data? (2) How do the use of both successful and failed rollouts, Context-Aware Corrective Matching, and the Adaptive Redirection Objective each contribute to RedFlow? (3) How does RedFlow compare to online RL methods in terms of sample efficiency? (4) Does RedFlow transfer to real-robot tasks and yield consistent policy improvement?

4.1 Experimental Setup

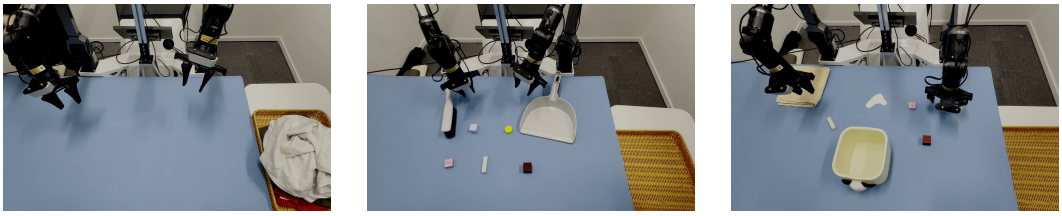
Simulation Experiments. We evaluate RedFlow on the four LIBERO suites [31]: Spatial, Object, Goal, and Long, each containing 10 tasks. We adopt π_0 [1] initialized from the π_{RL} [32] checkpoint as the base flow-matching VLA policy. Before offline RL fine-tuning, the base policy is trained on a pruned expert set $\mathcal{D}_{\text{exp}}$ containing 58 demonstrations for Spatial/Object/Goal and 208 demonstrations for Long. Offline RL fine-tuning uses $\mathcal{D} = \mathcal{D}_{\text{exp}} \cup \mathcal{D}_{\text{roll}}$, where $\mathcal{D}_{\text{roll}}$ contains 1,536 mixed-quality rollout trajectories per suite. Progress estimates are produced by a pretrained General Reward Model (GRM) [27], smoothed over temporal chunks, and combined with proprioceptive states to implement Context-Aware Corrective Matching via HDBSCAN-based context clustering [30]. All methods share the same training protocol within each task suite. We report average success rates over 500 evaluation episodes per suite, and full hyperparameters are provided in Appendix C.

Real-Robot Experiments. 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 with different manipulation demands: clothes folding, object sweeping, and table cleaning (Fig. 4), covering dexterous bi-manual manipulation, tool-mediated interaction, and pick-and-place manipulation, respectively. We initialize from the official π_0 base checkpoint provided by OpenPi [1] and obtain the task-specific base policies by fine-tuning on expert demonstrations for 50,000 steps per task. The expert demonstration sets contain 600 demonstrations for clothes folding, 200 for object sweeping, and 100 for table cleaning. For offline RL training, we collect 200, 100, and 100 rollouts from the corresponding base policies for clothes folding, object sweeping, and table cleaning, respectively, yielding mixed buffers of successes and failures. We report average success rates over 100 evaluation episodes per task. Full task configurations and training details are provided in Appendix D.

4.2 Main Results

Tab. 1 compares RedFlow with AWR [8] and DPO [33] on the four LIBERO suites. All methods share the same base policy, offline buffer $\mathcal{D}$, and evaluation protocol. AWR imitates rollout actions reweighted by estimated advantage, while DPO learns from trajectory-level preferences without constructing action-level corrective targets.

RedFlow achieves the best result on all four suites, lifting the average from 56.2% to **68.2%** (+12.0 points), outperforming AWR by 5.9 points and DPO by 8.5 points. The largest gain appears on LIBERO-Goal (71.2% vs. 57.8% / 51.8%), showing the benefit of explicitly localizing failure-inducing chunks and redirecting them toward corrective targets retrieved from similar contexts.



(a) Clothes Folding (b) Object Sweeping (c) Table Cleaning

Figure 4: Real-robot experiment setups. We evaluate RedFlow on a dual-arm Agilex Cobot Magic robot across three tasks requiring different manipulation skills: (a) clothes folding for dexterous bi-manual manipulation, (b) object sweeping for tool-mediated interaction, and (c) table cleaning for pick-and-place manipulation.

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	65.8	69.8	51.8	51.2	59.7
RedFlow (Ours)	**75.8**	**70.4**	**71.2**	**55.2**	**68.2**

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

Method	Spatial	Object	Goal	Avg
Rollout data composition				
w/o failure rollouts	71.4	67.4	65.2	68.0
w/o success rollouts	64.4	65.8	57.0	62.4
Context-Aware Corrective Matching				
w/o uncorrectable-failure separation	66.4	65.8	50.8	61.0
Adaptive Redirection Objective				
w/o $\mathcal{L}_{\text{cor}}$	70.4	68.4	68.6	69.1
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**

4.3 Ablation Studies

Tab. 2 ablates the three components of RedFlow in order: (i) rollout data composition, (ii) Context-Aware Corrective Matching, and (iii) the Adaptive Redirection Objective. Each variant removes one component while keeping the rest fixed.

(i) Rollout data composition. The first block shows that successes and failures provide complementary signals. Removing failures drops the average to 68.0%, since the policy loses the negative evidence required for suppression and redirection; removing successes drops it further to 62.4%, since neither high-quality anchors nor corrective targets remain. Combining both is needed to reach 72.5%.

(ii) Context-Aware Corrective Matching. The second block tests whether corrective targets should be assigned indiscriminately. Without uncorrectable-failure separation, every failure chunk receives a target regardless of whether a similar successful context exists, causing the largest drop in the table (72.5% $\rightarrow$ 61.0%, with a 20.4-point loss on LIBERO-Goal).

(iii) Adaptive Redirection Objective. The third block isolates the two failure-side terms. Removing $\mathcal{L}_{\text{sup}}$ alone (68.8%) leaves no mechanism to push probability mass away from failure-inducing actions; removing $\mathcal{L}_{\text{cor}}$ alone (69.1%) removes the redirection signal toward retrieved corrective targets. Removing both reduces the objective to quality-weighted attraction and drops the average to 65.7%. Suppression and correction are therefore complementary rather than redundant: the former excludes failure-prone regions, the latter specifies where probability mass should move.

4.4 Sample Efficiency Analysis

We compare the sample efficiency of RedFlow against three on-policy RL baselines—PPO [34], GRPO [35], and DDPO [36]—on LIBERO-Spatial. As shown in Fig. 5, RedFlow reaches 75.8% success (red dashed line) using only 1,536 offline trajectories and no additional environment interaction. The on-policy baselines collect 1,024 fresh rollouts at every update step; reaching the same success level requires roughly 13 / 16 / 24 update steps for PPO / GRPO / DDPO, corresponding to

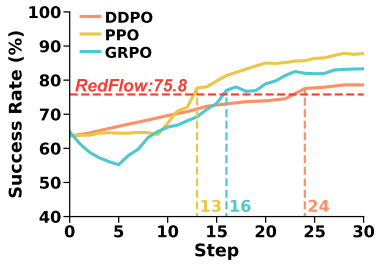


Figure 5: Sample efficiency comparison on LIBERO-Spatial against on-policy RL baselines.

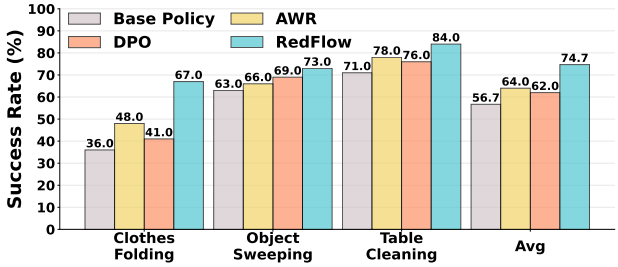


Figure 6: Success rates on real-robot tasks.

approximately 13K / 16K / 24K rollout trajectories—an order of magnitude more than RedFlow. This shows that structured failure reuse can substitute for a substantial portion of on-policy interaction.

4.5 Real-Robot Results

Fig. 6 reports success rates on the three real-robot tasks. RedFlow achieves the highest average success rate of 74.7%, outperforming all baselines on every task. The largest gain appears on clothes folding, where RedFlow improves the base policy from 36.0% to 67.0%, a 31.0-point absolute improvement that confirms the framework transfers to challenging bimanual manipulation.

Qualitative Recovery Analysis. Fig. 7 illustrates how RedFlow learns recovery behavior on clothes folding. The base policy fails when the T-shirt falls out of the right arm's reach: the right arm keeps attempting to grasp it but cannot recover. RedFlow instead executes a corrective retry—using the left arm to pull the T-shirt back into a reachable configuration before resuming the fold. This behavior emerges purely from offline buffer reuse: Context-Aware Corrective Matching retrieves left-arm pulling actions from similar progress–state contexts as corrective targets, and the Adaptive Redirection Objective redirects the failed grasp attempts toward them.

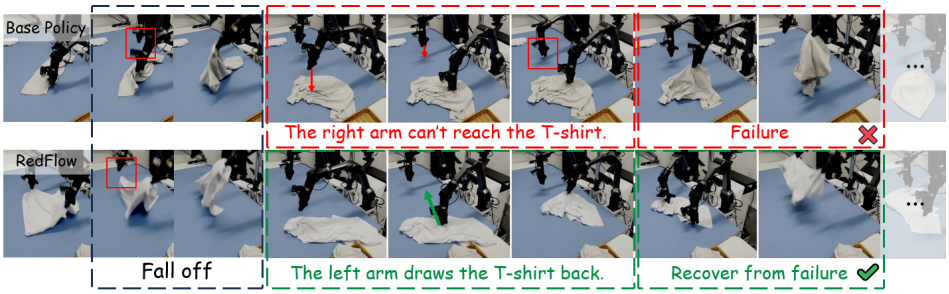


Figure 7: Qualitative correction behavior on clothes folding. The base policy fails when the T-shirt falls out of the right arm's reach, while RedFlow recovers by using the left arm to pull the cloth back and continue folding.

5 Conclusion

We presented RedFlow, an offline post-training framework that converts deployment failures of flow-matching VLA policies into action-level corrective supervision. Context-Aware Corrective Matching identifies recoverable failure-inducing chunks and retrieves successful alternatives from similar execution contexts, while the Adaptive Redirection Objective integrates these signals by reinforcing high-quality actions, suppressing undesirable ones, and redirecting recoverable failures toward retrieved targets. On LIBERO and three real-robot tasks, RedFlow surpasses offline baselines by clear margins and matches strong on-policy methods with roughly an order of magnitude fewer trajectories, lifting real-world success from 56.7% to 74.7%. These results establish structured failure reuse as a sample-efficient direction for post-training generalist VLA policies. Future work will investigate uncertainty-aware target assignment for out-of-distribution states, longer-horizon tasks, and extensions to iterative offline–online improvement.

References

[1] 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*, 2024.

[2] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL `https://arxiv.org/abs/2504.16054`.

[3] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL `https://proceedings.mlr.press/v229/zitkovich23a.html`.

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

[5] 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`.

[6] 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`.

[7] 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.

[8] 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`.

[9] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[10] 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, 2026. URL `https://arxiv.org/abs/2602.05049`.

[11] 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.

[12] 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`.

[13] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[14] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

[15] Wenke Xia, Yichu Yang, Hongtao Wu, Xiao Ma, Tao Kong, and Di Hu. Human-assisted robotic policy refinement via action preference optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pages 2679–2713. PMLR, 2025.

[17] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

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

[19] Quanxin Shou, Fangqi Zhu, Shawn Chen, Puxin Yan, Zhengyang Yan, Yikun Miao, Xiaoyi Pang, Zicong Hong, Ruikai Shi, Hao Huang, Jie Zhang, and Song Guo. Halo: A unified vision-language-action model for embodied multimodal chain-of-thought reasoning, 2026. URL `https://arxiv.org/abs/2602.21157`.

[20] Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.09515*, 2025.

[21] 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. *arXiv preprint arXiv:2509.15965*, 2025.

[22] 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.

[23] Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Rl-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint arXiv:2510.14830*, 2025.

[24] Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

[25] Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics, 2026. URL https://arxiv.org/abs/2601.00675.

[26] Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons, 2026. URL https://arxiv.org/abs/2603.02115.

[27] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-dopamine: General process reward modeling for high-precision robotic manipulation, 2025. URL https://arxiv.org/abs/2512.23703.

[28] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization, 2024. URL https://arxiv.org/abs/2402.01306.

[29] 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.

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

[31] 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, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

[32] 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, 2026. URL https://arxiv.org/abs/2510.25889.

[33] 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, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

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

[35] 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.

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

Appendix

A Bounded Corrective Redirection as Constrained Wasserstein Policy Transport

This appendix formalizes the endpoint-redirection principle stated in Theorem 1 and gives a broader variational interpretation of RedFlow from the perspective of conditional action-distribution transport. Rather than treating positive attraction, negative suppression, and corrective redirection as independent heuristics, we show that they arise as particle-level realizations of a local transport principle whose first variation yields a signed push–pull field. Throughout, $\mathcal{A} = \mathbb{R}^{K\times D}$ denotes the action-chunk space and $\|\cdot\|$ denotes the Frobenius norm. For any scalar r, we write $[r]_+ = \max\{r, 0\}$.

Scope. The theory characterizes the geometry of the RedFlow update conditional on the estimated chunk scores and assigned corrective targets. It establishes that the objective implements bounded exclusion from low-quality endpoints and constructive transport toward locally supported positive references.

Standing assumptions. We assume $\mathcal{A} = \mathbb{R}^{K\times D}$ with squared Euclidean cost, local behavior measures have finite second moments, soft gates are measurable and bounded in $(0, 1)$, and corrective targets are assigned only to clusters with non-empty estimated-positive support. Hinge derivatives are understood almost everywhere; the boundary $\|a - a^-\|^2 = m$ has zero measure under any absolutely continuous policy measure.

A.1 Conditional Action Measures and Soft Gating

For a local context z induced by the observation, instruction, and progress–state feature, let $\mu_\theta^z \in \mathcal{P}_2(\mathcal{A})$ denote the conditional action distribution induced by the flow policy. Let ν_β^z denote the local behavior distribution represented by the offline buffer. For the population-level transport energy, we use a smoothed empirical behavior reference $\nu_{\beta,\epsilon}^z$, which provides a regular density-level proxy for the local buffer distribution. In finite samples, $\nu_{\beta,\epsilon}^z$ is evaluated through the same local buffer points used by the practical objective.

The estimated chunk-level advantage induces a soft gate

$$w(a, z) = \sigma\big(\widehat{A}(a, z)/T_w\big), \tag{15}$$

which decomposes the local behavior reference into soft positive and negative components:

$$\mathrm{d}\nu_{+,w}^z(a) = \frac{w(a, z)}{Z_+^z}\,\mathrm{d}\nu_{\beta,\epsilon}^z(a), \qquad \mathrm{d}\nu_{-,w}^z(a) = \frac{1 - w(a, z)}{Z_-^z}\,\mathrm{d}\nu_{\beta,\epsilon}^z(a), \tag{16}$$

where

$$Z_+^z = \int_{\mathcal{A}} w(a, z)\,\mathrm{d}\nu_{\beta,\epsilon}^z(a), \qquad Z_-^z = \int_{\mathcal{A}} \big(1 - w(a, z)\big)\,\mathrm{d}\nu_{\beta,\epsilon}^z(a). \tag{17}$$

The sample-level weights $w_t = \sigma(\widehat{A}_t/T_w)$ and $1 - w_t$ used in RedFlow are Monte Carlo realizations of this soft measure decomposition.

A low-quality chunk is correctable only if it is assigned to a valid progress–state cluster whose estimated-positive subset is non-empty. For such a cluster $\mathcal{C}_c$, define

$$\mathcal{C}_c^+ = \{i \in \mathcal{C}_c : \widehat{A}_i > 0\}. \tag{18}$$

For uncorrectable low-quality chunks, no corrective target is defined and only bounded suppression is applied.

A.2 Corrective Targets as Advantage-Tilted Fréchet Means

Theorem 2 (Corrective target as empirical Fréchet mean)**.** *For a cluster $\mathcal{C}_c$ with non-empty positive subset $\mathcal{C}_c^+$, define*

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

Then b_z is the unique minimizer of the empirical weighted Fréchet problem

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

It is therefore the empirical squared-cost barycenter of the advantage-tilted positive support in the local progress–state cluster.

Proof. The weights satisfy $\alpha_i > 0$ and $\sum_i \alpha_i = 1$. The objective in Eq. (20) is strictly convex in the Euclidean action space. Its stationary condition is

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

which is the unique global minimizer. □

This empirical target is a plug-in estimator of an advantage-tilted local positive mean. Let the oracle advantage-tilted positive reference measure be

$$\nu_+^\star(a \mid z) \propto \nu_{\beta,+}(a \mid z) \exp(A^\star(z,a)/\kappa), \tag{22}$$

where $\nu_{\beta,+}$ is the positive component of the local behavior distribution and $A^\star$ is the latent local advantage. The squared-cost Fréchet mean of this tilted measure is

$$b_z^\star = \arg\min_{b\in\mathcal{A}} \int_{\mathcal{A}} \|b - a\|^2 \, \mathrm{d}\nu_+^\star(a \mid z). \tag{23}$$

Under the Euclidean action metric,

$$b_z^\star = \mathbb{E}_{a\sim\nu_+^\star(\cdot|z)}[a]. \tag{24}$$

In Euclidean action spaces, this expectation is exact for squared cost; on non-Euclidean action manifolds, the same construction generalizes to the corresponding manifold Fréchet mean. This explains why corrective redirection pulls toward a weighted positive barycenter rather than toward an arbitrary successful chunk.

A.3 A Local Transport Energy

We interpret local policy improvement as a proximal transport step over conditional action distributions. Building upon the Jordan–Kinderlehrer–Otto scheme, consider the one-step variational update

$$\mu_{k+1}^z = \arg\min_{\mu\in\mathcal{P}_2(\mathcal{A})} \frac{1}{2\tau} W_2^2(\mu, \mu_k^z) + \mathcal{E}_z(\mu), \tag{25}$$

where W_2 is the quadratic Wasserstein distance and $\mathcal{E}_z$ is the local transport energy

$$\begin{aligned}\mathcal{E}_z(\mu) = \frac{c_z \lambda_{\mathrm{cor}}}{2} &\int_{\mathcal{A}} \|a - b_z\|^2 \, \mathrm{d}\mu(a) \\ &+ \lambda_{\mathrm{sup}} \int_{\mathcal{A}} \int_{\mathcal{A}} \left[m - \|a - a^-\|^2\right]_+ \mathrm{d}\nu_{-,w}^z(a^-) \, \mathrm{d}\mu(a) \\ &+ \frac{\lambda_{\mathrm{bc}}}{2} \int_{\mathcal{A}} \int_{\mathcal{A}} w(a^-, z) \|a - a^-\|^2 \, \mathrm{d}\nu_{\beta,\epsilon}^z(a^-) \, \mathrm{d}\mu(a).\end{aligned} \tag{26}$$

Here $c_z \in \{0, 1\}$ indicates whether a corrective target exists. The first term attracts the transported action distribution toward the advantage-tilted positive reference b_z. The second term treats low-quality chunks as finite-range obstacles rather than global repulsive charges. The third term is an empirical endpoint-anchoring energy: it keeps transported particles tied to high-quality behavior endpoints through a weighted quadratic transport cost. This particle-based anchoring form matches the squared endpoint regression structure used by flow-matching training.

As $\tau \to 0$, the proximal sequence formally induces a Wasserstein gradient flow

$$\partial_s \mu_s^z + \nabla_a \cdot (\mu_s^z u_s^z) = 0, \qquad u_s^z(a) = -\nabla_a \frac{\delta \mathcal{E}_z}{\delta \mu}(a), \tag{27}$$

where s denotes the policy-transport time, distinct from the flow-matching timestep n.

A.4 Push–Pull Velocity from First Variation

The local transport energy in Eq. (26) induces a transport velocity field over action space. Its first variation gives the desired push–pull structure.

Theorem 3 (Push–pull velocity field)**.** *For almost every $a \in \mathcal{A}$, the Wasserstein steepest-descent velocity induced by $\mathcal{E}_z$ is*

$$\begin{aligned} u^z(a) = c_z\lambda_{\mathrm{cor}}(b_z - a) \\ + 2\lambda_{\mathrm{sup}} \int_{\mathcal{A}} \mathbf{1}\{\|a - a^-\|^2 < m\}(a - a^-)\, \mathrm{d}\nu^z_{-,w}(a^-) \\ + \lambda_{\mathrm{bc}} \int_{\mathcal{A}} w(a^-, z)(a^- - a)\, \mathrm{d}\nu^z_{\beta,\epsilon}(a^-). \end{aligned} \tag{28}$$

Proof. The first variation of the corrective attraction term is

$$\frac{c_z\lambda_{\mathrm{cor}}}{2}\|a - b_z\|^2, \tag{29}$$

whose negative spatial gradient is $c_z\lambda_{\mathrm{cor}}(b_z - a)$.

The first variation of the obstacle term is

$$\lambda_{\mathrm{sup}} \int_{\mathcal{A}} \left[m - \|a - a^-\|^2\right]_+ \mathrm{d}\nu^z_{-,w}(a^-). \tag{30}$$

For almost every a, its negative spatial gradient is

$$2\lambda_{\mathrm{sup}} \int_{\mathcal{A}} \mathbf{1}\{\|a - a^-\|^2 < m\}(a - a^-)\, \mathrm{d}\nu^z_{-,w}(a^-). \tag{31}$$

The first variation of the empirical endpoint-anchoring term is

$$\frac{\lambda_{\mathrm{bc}}}{2} \int_{\mathcal{A}} w(a^-, z)\|a - a^-\|^2\, \mathrm{d}\nu^z_{\beta,\epsilon}(a^-), \tag{32}$$

whose negative spatial gradient is

$$\lambda_{\mathrm{bc}} \int_{\mathcal{A}} w(a^-, z)(a^- - a)\, \mathrm{d}\nu^z_{\beta,\epsilon}(a^-). \tag{33}$$

Combining the three components gives the stated velocity field. □

Theorem 3 gives the structural justification for RedFlow. Positive and negative samples do not enter symmetrically. Positive samples define constructive transport destinations through b_z, whereas negative samples define finite-range obstacles that exclude locally undesirable action regions. Therefore, the update is not pure repulsion from failures; it redirects probability mass from failure neighborhoods toward locally supported positive references while remaining anchored to the behavior support.

The margin m has a precise geometric meaning in this view. It specifies the interaction radius of the obstacle ball

$$\mathcal{B}(a^-, \sqrt{m}) = \{a : \|a - a^-\|^2 < m\}. \tag{34}$$

Inside this ball, the negative chunk induces a repulsive transport component. Outside it, the negative chunk exerts no force. This finite-range structure is essential in continuous action spaces: negative evidence identifies where probability mass should not remain, but does not specify a global direction in which the policy should move.

A.5 Flow-Matching Endpoint Realization

Theorem 4 (Endpoint attraction as velocity regression)**.** *Under the linear endpoint parameterization $\widehat{x}_0 = x_n - nv_\theta(x_n, n, z)$, endpoint attraction toward any $y \in \mathcal{A}$ is equivalent, up to the positive factor n^2, to flow-matching velocity regression toward the linear-path velocity $u^y_n(x_n, z) = (x_n - y)/n$.*

Proof. For an endpoint target $y \in \mathcal{A}$, define the corresponding linear-path target velocity as

$$u_n^y(x_n, z) = \frac{x_n - y}{n}. \tag{35}$$

Under the reverse-time endpoint parameterization used by the policy, the denoised prediction is

$$\widehat{x}_0 = x_n - n v_\theta(x_n, n, z). \tag{36}$$

Therefore,

$$\|\widehat{x}_0 - y\|^2 = n^2 \left\| v_\theta(x_n, n, z) - u_n^y(x_n, z) \right\|^2. \tag{37}$$

Note that while the minimizers are identical, the scale factor n^2 implies that regressing in the velocity space implicitly schedules the strength of endpoint attraction. This time-dependent scaling is a standard feature of flow-matching objectives, naturally down-weighting the effective endpoint constraint at early flow stages ($n \approx 1$, closer to noise) and strengthening it near the generation boundary ($n \approx 0$). □

Thus, endpoint-space attraction toward y can be implemented as target-velocity regression at the same noisy anchor. Choosing $y = a_t$ gives empirical behavior anchoring through standard flow-matching regression to observed buffer endpoints. Choosing $y = b_{z_t}$ gives corrective attraction toward the local positive reference. The hinge term acts as an endpoint-space obstacle potential around negative actions.

At the endpoint level, the suppression potential for a negative chunk a_t is

$$\phi_{\sup}(\widehat{x}_0, a_t) = [m - \|\widehat{x}_0 - a_t\|^2]_+. \tag{38}$$

For $\|\widehat{x}_0 - a_t\|^2 \neq m$, its negative endpoint gradient is

$$-\nabla_{\widehat{x}_0} \phi_{\sup} = 2\mathbf{1}\{\|\widehat{x}_0 - a_t\|^2 < m\}(\widehat{x}_0 - a_t). \tag{39}$$

Thus, the endpoint-level suppression direction pushes the prediction away from a_t only inside the margin and vanishes outside the obstacle ball.

Combining the transport components at the particle level yields the RedFlow endpoint surrogate

$$\begin{aligned} \mathcal{L}_t = {} & w_t \|v_\theta(x_n, n, z_t) - u_n^{a_t}(x_n, z_t)\|^2 \\ & + \lambda_{\sup}(1 - w_t)[m - \|\widehat{x}_0 - a_t\|^2]_+ \\ & + c_t \lambda_{\mathrm{cor}}(1 - w_t)\|\widehat{x}_0 - b_{z_t}\|^2. \end{aligned} \tag{40}$$

Here $c_t \in \{0, 1\}$ indicates whether the negative chunk has an assigned corrective target. When w_t is large, the sample primarily contributes to empirical behavior anchoring; when w_t is small, it primarily contributes to finite-range suppression and, if $c_t = 1$, corrective redirection.

The first term is the flow-matching realization of the empirical endpoint-anchoring energy. For a sampled behavior endpoint a_t, the endpoint quadratic cost $w_t\|\hat{x}_0 - a_t\|^2$ induces the attraction direction $w_t(a_t - \hat{x}_0)$. Under the linear endpoint parameterization $\hat{x}_0 = x_n - n v_\theta(x_n, n, z_t)$ and $u_n^{a_t}(x_n, z_t) = (x_n - a_t)/n$, we have

$$\|\hat{x}_0 - a_t\|^2 = n^2 \|v_\theta(x_n, n, z_t) - u_n^{a_t}(x_n, z_t)\|^2. \tag{41}$$

Thus, weighted velocity regression is the flow-matching form of weighted endpoint attraction. The second term realizes finite-range endpoint exclusion around low-quality chunks, while the third term realizes corrective attraction toward the advantage-tilted positive reference. Together, Eq. (40) instantiates the push–pull transport geometry in Theorem 3 at the particle level.

A.6 Energy Dissipation and Local Stationary Convergence

While establishing global convergence for non-linear deep neural networks over continuous action spaces is generally intractable, the Wasserstein gradient flow formulation provides a strong geometric guarantee of local stationary convergence for the RedFlow update.

Theorem 5 (Energy Dissipation and Stationary Convergence)**.** *Let μ_s^z be the solution to the idealized Wasserstein gradient flow $\partial_s \mu_s^z + \nabla_a \cdot (\mu_s^z u_s^z) = 0$ driven by the local transport energy $\mathcal{E}_z(\mu)$ in Eq.* (26)*. Assume the initial energy is finite. Then the energy is strictly non-increasing along the flow:*

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{E}_z(\mu_s^z) = -\int_{\mathcal{A}} \|u_s^z(a)\|^2 \,\mathrm{d}\mu_s^z(a) \le 0. \tag{42}$$

Consequently, under this non-parametric particle dynamics, the distribution converges to a stationary configuration where the push-pull velocity field vanishes almost everywhere.

Proof. By the chain rule for Wasserstein gradient flows, the time derivative of the energy functional along its own gradient flow is given by the negative squared $L^2(\mu_s^z)$ norm of the minimal-norm subdifferential. Since $u_s^z(a) = -\nabla_a \frac{\delta \mathcal{E}_z}{\delta \mu}(a)$, we have:

$$\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{E}_z(\mu_s^z) = \int_{\mathcal{A}} \frac{\delta \mathcal{E}_z}{\delta \mu}(a)\partial_s \mu_s^z(a)\,\mathrm{d}a = -\int_{\mathcal{A}} \left\| \nabla_a \frac{\delta \mathcal{E}_z}{\delta \mu}(a) \right\|^2 \mathrm{d}\mu_s^z(a) \le 0. \tag{43}$$

The local transport energy $\mathcal{E}_z(\mu)$ is bounded from below by 0, as it is a pure sum of non-negative quadratic distances and non-negative hinge losses (Eq. (26)). Because $\mathcal{E}_z$ is bounded from below and monotonically decreasing, the limit $\lim_{s\to\infty} \mathcal{E}_z(\mu_s^z)$ exists and is finite. Consequently, the dissipation rate must decay to zero:

$$\lim_{s\to\infty} \int_{\mathcal{A}} \|u_s^z(a)\|^2 \,\mathrm{d}\mu_s^z(a) = 0. \tag{44}$$

This implies that the probability mass strictly stabilizes, converging to a stationary distribution μ_∞^z where the constructive attraction and finite-range suppression forces perfectly balance. □

This theorem provides the geometric safeguard for RedFlow. While practical deep neural network optimization involves complexities like finite capacity and mini-batch stochasticity that preclude absolute convergence guarantees, this continuous-time particle view ensures that the underlying objective is intrinsically dissipative. It demonstrates that despite the non-convexity introduced by the finite-range obstacle term ($\mathcal{L}_{\text{sup}}$), the fundamental transport forces are designed to optimally balance behavioral anchoring, physical safety constraints, and constructive redirection without inherent oscillatory dynamics.

A.7 Proof of the Endpoint Redirection Principle

We now prove the single-obstacle endpoint result used in Theorem 1. This result characterizes the interaction between bounded suppression and target-guided correction before considering the population-level transport view.

Theorem 6 (Single-obstacle endpoint geometry)**.** *Consider a single negative obstacle $a^- \in \mathcal{A}$, a corrective target $b_z \in \mathcal{A}$, and an endpoint variable $h \in \mathcal{A}$. Assume $b_z \neq a^-$, $m > 0$, and $\lambda_{\text{sup}} \ge \lambda_{\text{cor}} > 0$. The minimizer of*

$$\phi(h) = \lambda_{\text{cor}}\|h - b_z\|^2 + \lambda_{\text{sup}}[m - \|h - a^-\|^2]_+ \tag{45}$$

coincides with the projection of b_z onto the complement of the obstacle ball:

$$h^\star = \begin{cases} b_z, & \|b_z - a^-\|^2 \ge m, \\ a^- + \sqrt{m}\,\dfrac{b_z - a^-}{\|b_z - a^-\|}, & \|b_z - a^-\|^2 < m. \end{cases} \tag{46}$$

Equivalently, $h^\star$ solves the constrained projection problem

$$\min_{h\in\mathcal{A}} \|h - b_z\|^2 \qquad \text{s.t.} \qquad \|h - a^-\|^2 \ge m. \tag{47}$$

Proof. If $\|b_z - a^-\|^2 \ge m$, the hinge term is inactive at $h = b_z$, and $h = b_z$ minimizes both terms in Eq. (45). Now suppose $d = \|b_z - a^-\| < \sqrt{m}$. For fixed $r = \|h - a^-\|$, the hinge term depends only on r, and the attraction term is minimized on the ray

$$h = a^- + r\frac{b_z - a^-}{d}. \tag{48}$$

Indeed, among points at radius r from a^-, this ray point maximizes the inner product with $b_z - a^-$ by Cauchy–Schwarz and therefore minimizes $\|h - b_z\|^2$; no off-ray point can achieve a smaller objective. Inside the obstacle ball, the problem therefore reduces to the one-dimensional objective

$$\psi(r) = \lambda_{\mathrm{cor}}(d - r)^2 + \lambda_{\mathrm{sup}}(m - r^2), \qquad 0 \le r < \sqrt{m}. \tag{49}$$

Its derivative is

$$\psi'(r) = 2(\lambda_{\mathrm{cor}} - \lambda_{\mathrm{sup}})r - 2\lambda_{\mathrm{cor}}d. \tag{50}$$

Since $\lambda_{\mathrm{sup}} \ge \lambda_{\mathrm{cor}} > 0$ and $d > 0$, we have $\psi'(r) < 0$ for all $0 \le r < \sqrt{m}$. Hence the objective decreases up to the boundary $r = \sqrt{m}$. Outside the obstacle ball, the hinge term vanishes, and the closest feasible point to b_z is precisely the boundary projection in Eq. (46). □

Consequently, hinge suppression implements bounded exclusion, while the correction term selects the closest safe endpoint aligned with the local positive reference. The endpoint subsystem therefore does not induce arbitrary repulsion from failures; it redirects the prediction toward the nearest margin-satisfying point consistent with the corrective target. This proves the endpoint principle summarized in Theorem 1.

Theorem 7 (Bounded redirection of the corrective subsystem)**.** *For a correctable negative chunk with $c_t = 1$ and fixed weight $w_t < 1$, consider the corrective subsystem of the sample objective, isolated by analyzing only the terms acting away from the failure anchor:*

$$\lambda_{\mathrm{sup}}(1 - w_t)[m - \|\widehat{x}_0 - a_t\|^2]_+ + \lambda_{\mathrm{cor}}(1 - w_t)\|\widehat{x}_0 - b_{z_t}\|^2$$

This subsystem shares the exact same minimizers as Theorem 6 with $a^- = a_t$ and $b_z = b_{z_t}$. Hence, the repulsive force vanishes outside the margin ball around a_t, and inside the margin, the correction term selects the closest margin-satisfying endpoint aligned with the local positive reference. The final prediction naturally interpolates between this bounded corrective target and the empirical anchor a_t governed by the omitted attraction term $\mathcal{L}_{\mathrm{att}}$.

Proof. The positive scalar $(1 - w_t)$ does not change minimizers. The remaining objective is exactly Eq. (45) after substituting $a^- = a_t$ and $b_z = b_{z_t}$. □

A.8 Interpretation

The analysis characterizes RedFlow as a local transport procedure in continuous action space. Positive chunks provide constructive destinations, negative chunks provide finite-range exclusion regions, and behavior anchoring keeps the transported distribution tied to the local buffer support. Flow matching provides the velocity-parameterized mechanism for realizing the resulting endpoint transport. In multimodal neighborhoods, the same framework can be extended by replacing the single barycentric target with mixture-valued positive references.

B Training Details

Overall training pipeline. RedFlow follows a single-iteration offline post-training pipeline for flow-based 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 $8\times$H20 GPUs. We then collect a fixed offline rollout buffer with the base policy, including both successful and failed trajectories. Given the frozen buffer, we estimate chunk-level progress with a pretrained GRM, assign action-level quality labels, construct corrective targets with progress–state-aware HDBSCAN clustering, and finally update the policy using the adaptive redirection objective.

Progress estimation and corrective target assignment. For both simulation and real-robot experiments, we use the pretrained GRM [27] to estimate task progress and apply the progress–state-aware corrective targeting procedure described in Section 3.2. Progress estimates are smoothed over temporal chunks and combined with trajectory outcomes to obtain the proxy advantage $\hat{A}_t$. We then cluster action chunks with HDBSCAN [30] in the normalized progress–state space and assign corrective targets from co-clustered positive chunks when available. Detailed HDBSCAN and progress-estimation hyperparameters are provided separately for simulation and real-robot experiments in Tables 4 and 6.

Offline policy optimization. The policy is optimized with the adaptive redirection objective in Eq. (14), which combines quality-weighted attraction, failure suppression, and target-guided correction. 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: Pretrained policy π_θ, optional demonstrations $\mathcal{D}_{\text{exp}}$, GRM R, task instruction l, epochs E

// Phase 1: Data collection

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

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

// Phase 2: Progress-state Clustering for corrective targets

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

  $\bar{p}_t \leftarrow \frac{1}{2W+1} \sum_{j=t-W}^{t+W} R(o_j, l)$

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

  $c_t \leftarrow 0$

end for

$\{\mathcal{C}_c\} \leftarrow \text{HDBSCAN}(\{f_t\}), \quad f_t = [\,\tilde{q}_t;\, \beta\,\bar{p}_t\,]$

for a_t with $\hat{A}_t < 0$ **do**

  Let $\mathcal{C}_c$ be the HDBSCAN 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

// Phase 3: Adaptive redirection

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

  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ using Eq. (14)

end for

return π_θ

C Simulation Implementation Details

Benchmark and base policy. We evaluate RedFlow on the four LIBERO suites [31]: Spatial, Object, Goal, and Long, each containing 10 tasks. We adopt π_0 [1] initialized from the πRL [32] checkpoint as the base VLA policy. The SFT policy is trained on a pruned expert set $\mathcal{D}_{\text{exp}}$ containing 58 demonstrations for Spatial, Object, and Goal, and 208 demonstrations for Long.

Offline RL data. RL fine-tuning uses

$$\mathcal{D} = \mathcal{D}_{\text{exp}} \cup \mathcal{D}_{\text{roll}}, \tag{51}$$

where $\mathcal{D}_{\text{roll}}$ contains 1,536 mixed-quality rollout trajectories per suite. Progress estimates are produced by the pretrained GRM, smoothed over temporal chunks, and combined with proprioceptive states for HDBSCAN-based corrective target assignment. 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.

Hyperparameter choices for LIBERO. Tab. 3 summarizes the offline training configuration used across all four LIBERO suites, and Tab. 4 reports the HDBSCAN parameters used for corrective target assignment. 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.

Table 3: 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
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 action-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 $a_t^\star$.

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 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.

Table 4: HDBSCAN hyperparameters for progress–state-aware corrective target assignment 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 for advantage-weighted softmax over positive samples in the same cluster.

Table 5: 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.`

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: cloth folding, object sweeping, and table cleaning, covering dexterous bi-manual 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 [1]. 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 cloth folding, 200 for object sweeping, and 100 for table cleaning. For offline RL training, we collect 200, 100, and 100 rollouts from the initial policy for cloth folding, object sweeping, and table cleaning, respectively, yielding a mixed buffer of successes and failures.

Progress estimation and corrective target assignment. For real-robot experiments, RoboDopamine/GRM progress predictions are computed with `frame_interval=10`. The HDBSCAN-based corrective target assignment uses `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.

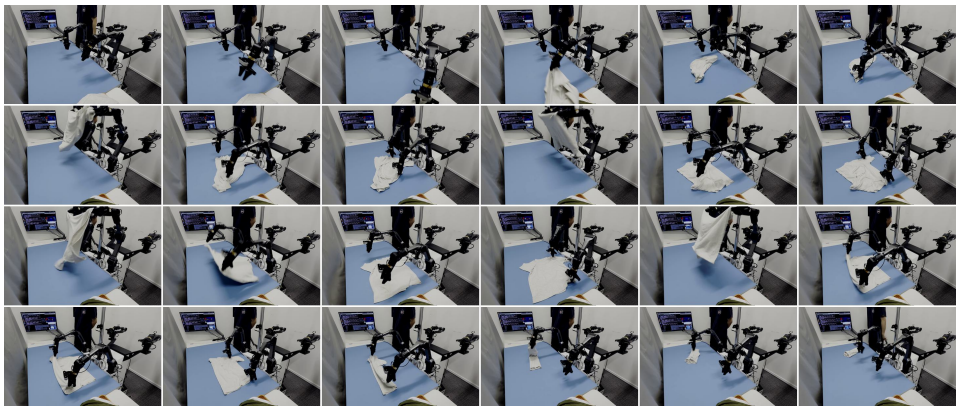


Figure 8: Successful trajectory of cloth folding.

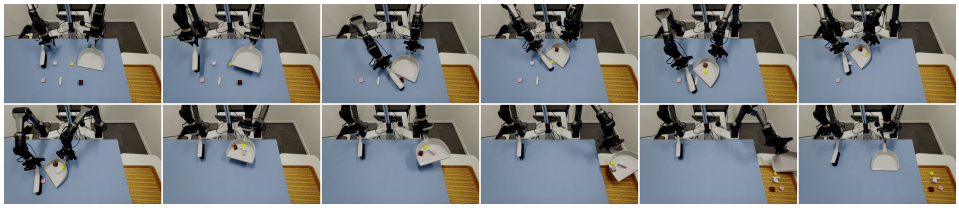


Figure 9: Successful trajectory of object sweeping.

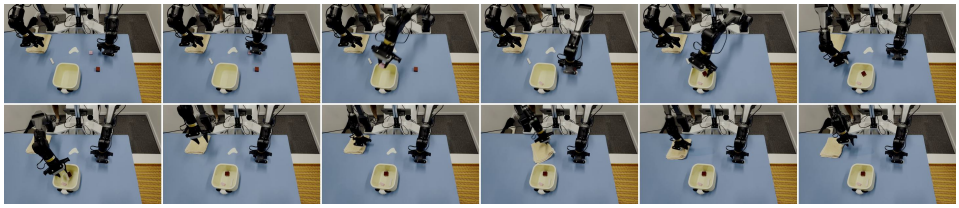


Figure 10: Successful trajectory of table cleaning.

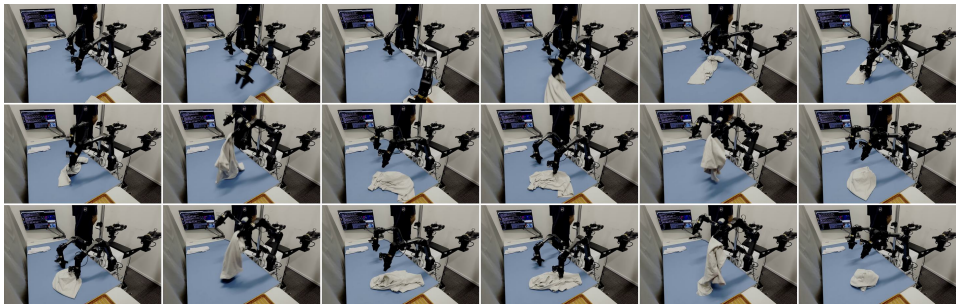


Figure 11: Failed trajectory of cloth folding.

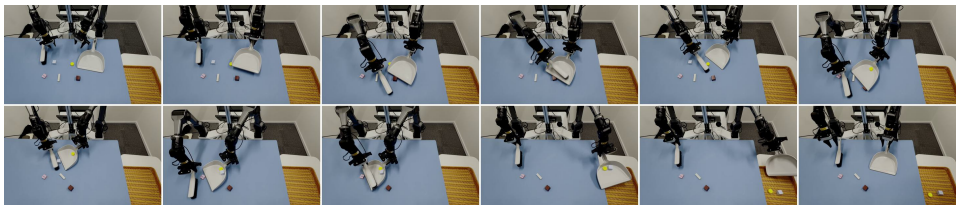


Figure 12: Failed trajectory of object sweeping.

Figure 13: Failed trajectory of table cleaning.

Table 6: Hyperparameters and setup for real-world experiments. All three tasks share the same π_0 LoRA backbone, adaptive redirection objective, optimizer, and distributed training configuration; task-specific items are listed per task.

Hyperparameter	**Cloth 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-finetuned 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		
Adaptive redirection objective			
Soft-weight temperature, T_w	3.0		
λ_{sup}	0.3		
Adaptive margin scale for m	1.0		
λ_{cor}	0.3		
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

E Diagnostics of Corrective-Target Assignment

Table 7: Diagnostics of negative-chunk usage on LIBERO. All negative chunks are used for suppression; only negative chunks whose progress–state cluster contains positive chunks additionally receive target-guided correction. We report the fraction optimized with suppression only, the fraction optimized with suppression plus correction, and the mean proxy advantage of the positive chunks used to construct each target, $\hat{A}^\star = \sum_{i \in C_c^+} \alpha_i \hat{A}_i$.

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	$+0.243$

F Limitations

While RedFlow demonstrates strong improvements in both simulation and real-world manipulation tasks, it still has several limitations. First, its corrective target assignment depends on the quality of task-progress estimates and progress–state clustering; inaccurate progress signals or noisy clustering may lead to imperfect failure-to-success matching. Second, RedFlow can only provide constructive redirection for failures that have nearby positive support in the offline buffer. Failure modes that are novel, out-of-distribution, or lack successful counterparts are only suppressed rather than explicitly corrected. Finally, our experiments focus on LIBERO and three real-robot tasks with fixed embodiments and settings, and broader validation across more diverse robots, visual conditions, and long-horizon tasks remains an important direction for future work.