T-REX-RL: IMPROVING TACTILE-REACTIVE DEXTEROUS MANIPULATION WITH REAL-WORLD REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

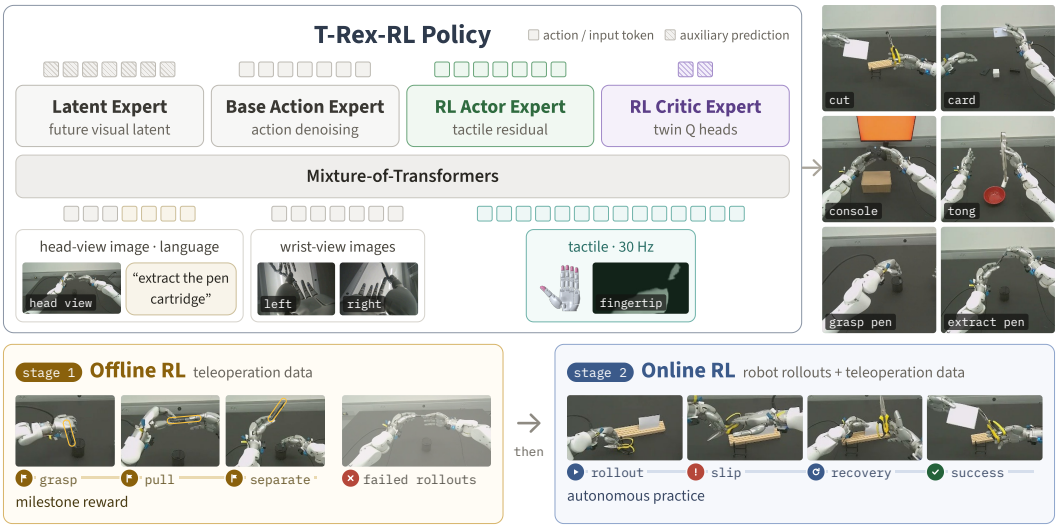


Figure 1: **T-Rex-RL overview.** T-Rex-RL brings real-world reinforcement learning to tactile-reactive VLAs. Tactile RL actor and critic experts refine the VLA's actions: offline RL on teleoperation data, then online RL on autonomous rollouts mixed with teleoperation data.

ABSTRACT

Tactile-reactive vision-language-action (VLA) models combine fast closed-loop tactile control with the broad visuomotor priors acquired through large-scale pretraining. However, existing models are mainly trained with imitation learning on teleoperated demonstrations, limiting their ability to improve beyond the quality and coverage of offline data. In this work, we study how tactile-reactive VLAs can improve via real-world interaction using offline-to-online reinforcement learning (RL). To achieve this, we initialize a tactile-reactive actor and critic through offline RL with teleoperation data, then continuously update them using online robot rollouts. The VLA is pretrained on large-scale data, whereas RL post-trains it on limited task-specific robot experience. We therefore freeze the VLA and add the RL actor and critic as Mixture-of-Transformers (MoT) experts that attend to its representations. While the VLA predicts action plans at a low frequency, the RL actor refines them at a high frequency from tactile signals to react to contact changes. The critic evaluates the refined actions to guide the actor's learning. We evaluate our approach on a suite of long-horizon bimanual dexterous manipulation tasks, where our method substantially outperforms baselines.

1 INTRODUCTION

Tactile sensing is widely recognized as crucial for dexterous manipulation, as it captures force variations, micro-slips, and local deformations that cameras cannot observe. Recent tactile-reactive vision-language-action (VLA) models (Huang et al., 2025; Bi et al., 2026; Cheng et al., 2026; Niu

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.

et al., 2026) bring fast closed-loop tactile control to large-scale pretrained visuomotor policies, enabling them to handle contact-rich tasks such as sliding a card or cutting paper. However, these models are trained with imitation learning on teleoperated demonstrations, which are not only prohibitively expensive to collect at scale, but also limit the policy's performance to the quality and coverage of the collected data.

Reinforcement learning (RL) offers a natural way to go beyond the training distribution. A growing body of work shows that VLAs can improve through real-world deployments via RL (Amin et al., 2026; Xu et al., 2026a; Chen et al., 2025b), using autonomously collected experience to correct deployment-time failures and to improve speed and robustness beyond the level of human teleoperation. These works, however, largely focus on vision-driven policies. How to apply real-world RL to tactile-reactive VLAs remains an open question.

We present T-Rex-RL, an offline-to-online RL framework for tactile-reactive VLAs. T-Rex-RL learns high-frequency tactile-reactive action refinement with RL, running asynchronously to the base-action expert's low-frequency action prediction. An RL actor predicts a residual on each plan from fresh tactile signals, and an RL critic evaluates the refined action. While the VLA is pretrained on large-scale data, RL post-trains it on limited robot experience, so we make the design asymmetric: the VLA experts stay frozen, and only the actor and critic, added as Mixture-of-Transformers experts (Liang et al., 2025), are trained. The actor and critic are first trained with offline RL on teleoperation data, and then continuously updated on autonomous rollouts.

We evaluate on five long-horizon bimanual dexterous manipulation tasks with tactile-sensing hands. With approximately 30 minutes of online robot execution per task, excluding human-assisted resets, T-Rex-RL improves the average task-progress score by 35 percentage points over its offline-only variant on the three-task ablation suite (Table 2) and by 22 percentage points over the T-Rex imitation-learning baseline across all five tasks (Table 1). Our ablations further assess the contributions of tactile feedback, asynchronous action refinement, and online RL.

We summarize our contributions as follows: (i) We bring real-world RL to tactile-reactive VLAs, enabling them to improve from their own deployments without additional teleoperation; (ii) We propose a residual actor-critic parameterized as additional MoT experts, reusing the VLA's multimodal representations and trained offline-to-online; (iii) We validate the approach on five long-horizon, real-world bimanual dexterous tasks, with ablations on the architecture and training recipe.

2 Related Work

2.1 Real World RL

Real-world RL improves robotic policies directly through physical interaction (Kober et al., 2013; Ibarz et al., 2021). Learning on real hardware avoids the gap between simulated and real contact dynamics, which otherwise requires extensive randomization (Andrychowicz et al., 2020) and can still cause zero-shot transfer to fail (Yin et al., 2025). In exchange, exploration, rewards, and resets become practical bottlenecks, since every trial costs robot time and human supervision (Singh et al., 2019; Zhu et al., 2020). Early work first showed that deep neural network policies can be trained directly on real robots (Gu et al., 2017; Levine et al., 2015; 2016) while later work scaled this idea by collecting experience in parallel on several robots, learning vision-based grasping and other multi-task manipulation skills (Kalashnikov et al., 2018; 2022).

Because such scale is expensive, later work reuses prior knowledge to learn from fewer trials (Zhu et al., 2019; Schoettler et al., 2020). Demonstrations can guide exploration when rewards are sparse (Vecerik et al., 2017), and residual RL learns only a correction on top of an existing controller (Johannink et al., 2019). Offline-to-online methods make fuller use of prior data, either pretraining on it or replaying it during online learning (Nair et al., 2020; Nakamoto et al., 2023; Ball et al., 2023). Combining these ideas, recent systems achieve near-perfect success on precise tasks, such as circuit-board insertion and cable routing, from modest real-world interaction (Luo et al., 2024; 2025; Lei et al., 2026). We follow this line and bring offline-to-online residual RL to pretrained tactile-reactive VLAs.

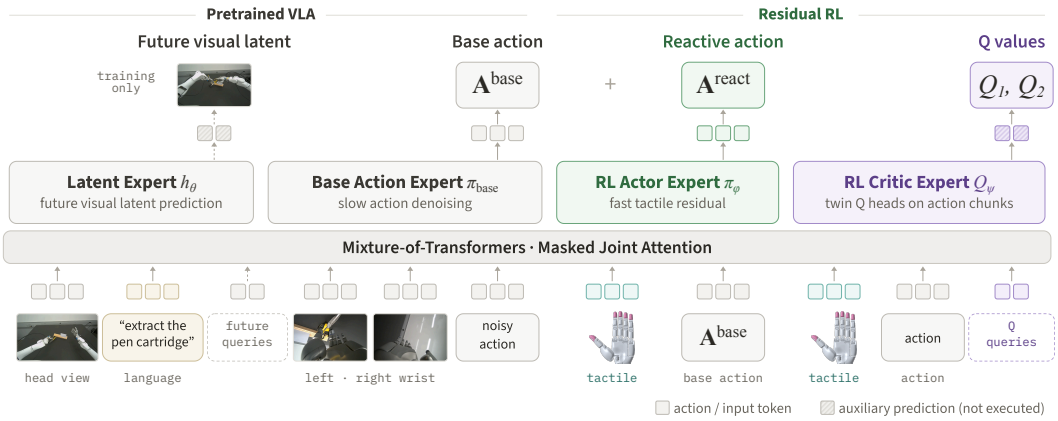


Figure 2: **T-Rex-RL architecture.** The latent and base-action experts supply visual-language context and a denoised 16-step action proposal. Tactile-conditioned RL experts read their cached representations: the actor refines the proposal at offsets $0, 4, 8, 12$, and the critic evaluates full action chunks through two Q heads.

2.2 RL FOR VLA

VLAs pretrained on large and diverse robot data can perform many manipulation skills out of the box (Black et al., 2025b), but supervised fine-tuning on limited and inconsistent demonstrations struggles to reach robust performance, especially in contact-rich tasks (Chen et al., 2025b). RL addresses this limit by letting the VLA improve from its own deployments, increasing both success rate and task throughput (Amin et al., 2026). Existing methods differ mainly in how the RL signal reaches the VLA. PLD (Xiao et al., 2026) uses residual RL to collect recovery trajectories and distills them back into the VLA, whereas πRL (Chen et al., 2025a) fine-tunes the VLA weights directly with online RL. BORA (Chen et al., 2026b) instead keeps the VLA frozen and learns an online residual on top of it, guided by an offline action-conditioned critic and human interventions.

Closest to ours, two methods refine VLA actions online. Real-Time EXPO-FT (Dong et al., 2026) edits action chunks from fresh observations to handle latency, while TORL-VLA (Zheng et al., 2026) uses tactile wrench feedback and an intervention-censored critic in single-arm gripper manipulation. T-Rex-RL extends this to bimanual dexterous hands, refining asynchronously with tactile actor–critic MoT experts trained offline-to-online.

2.3 TACTILE REACTIVE POLICIES FOR DEXTEROUS MANIPULATION

Tactile sensing is increasingly used in dexterous manipulation: HATO (Lin et al., 2025) learns bimanual visuotactile skills from demonstrations, SaTA (Huang et al., 2026) anchors tactile observations in the hand's kinematic frame, FingerEye (Xu et al., 2026b) integrates sensing across pre-contact and contact phases, and ViTacFormer (Heng et al., 2026) learns cross-modal representations for dexterous imitation.

Reactive Diffusion Policy (RDP) (Xue et al., 2025) combines slow action-chunk prediction with fast tactile refinement. T-Rex (Niu et al., 2026) extends this separation with large-scale tactile pretraining and variable-rate experts. These imitation-learning approaches complement the visuomotor priors of VLAs such as $\pi_{0.5}$ (Black et al., 2025a) with contact feedback.

For contact-aware RL, Visuotactile-RL (Hansen et al., 2022) studies visual–tactile learning in simulation, ContactExplorer (Liu et al., 2026) uses contact-guided exploration and sim-to-real transfer, and ViTaL (Zhao et al., 2025) learns local visuotactile skills through residual RL. T-Rex-RL improves a pretrained tactile-reactive VLA through offline-to-online RL on real-world bimanual dexterous tasks.

3 METHOD

T-Rex-RL adds residual RL to a pretrained tactile-reactive VLA (Figure 2). High-level action commands are issued at 30 Hz. The VLA takes language, camera images, and proprioception as input and proposes a 16-step action chunk (Zhao et al., 2023). Two RL experts share its Mixture-of-Transformers backbone: an actor that refines the same proposal using fresh tactile feedback at intra-chunk offsets $0, 4, 8, 12$, and a critic that scores full action chunks. We describe the architecture (Section 3.1), the offline-to-online RL objective (Section 3.2), the training recipe (Section 3.3), and real-time execution (Section 3.4).

3.1 ACTOR–CRITIC ARCHITECTURE WITH MIXTURE-OF-TRANSFORMER-EXPERTS

The policy consists of four Mixture-of-Transformer-Experts (MoT) streams: latent (h_θ), base-action (π_{base}), RL actor (π_ϕ), and RL critic (Q_ψ) (Figure 2). Each stream has separate attention projections and feed-forward parameters, with masked attention providing access to other streams' representations. We initialize the visuomotor backbone and reuse the tactile encoding design from T-Rex (Niu et al., 2026).

Predictive visuomotor backbone. The latent and base-action experts learn a visuomotor prior from human and robot demonstrations. The latent expert encodes the head image and instruction. Following FLARE (Zheng et al., 2025), 32 training-only queries predict features of 8 future frames, aligned to the frozen vision encoder's embeddings rather than decoded into pixels. The base-action expert attends to the latent tokens and wrist images, and generates action chunks from Gaussian noise by flow matching.

Contact-conditioned actor and critic. The RL experts integrate high-rate tactile signals while preserving the visuomotor prior. Both visuomotor experts therefore stay frozen during RL; their cached keys and values let each refinement reuse the backbone without recomputing it. Each fingertip senses force/torque and deformation; we encode its force/torque history, current force, and deformation map. The actor corrects the base proposal, and the critic evaluates the corrected action chunk with two Q heads corresponding to two query tokens.

Strided RL experts. We design the RL experts to be efficient for online training while still leveraging the backbone's deep representations. Instead of a full-depth expert, each RL expert has 4 blocks aligned with backbone layers $\{7, 14, 21, 28\}$, about 200M parameters (Figure 3). Each block reads the cached prefix at its layer and passes its tokens to the next RL block, so a short trainable stream sees progressively deeper representations without duplicating the backbone.

With all experts sharing one backbone, the attention mask prevents interference among them: the prefix is read-only, and the actor and critic streams cannot attend to each other. Training the critic therefore leaves the actor and backbone unchanged, and training the actor uses only the critic's gradient with respect to the action. At deployment, only the actor runs. See Appendix A.6 for details.

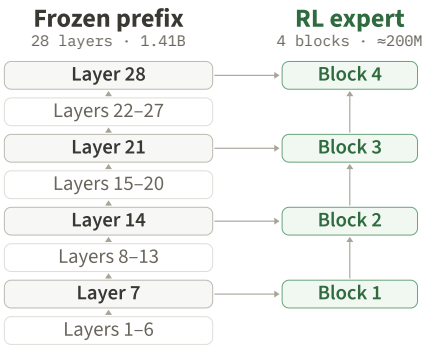


Figure 3: **Strided RL experts.** Each RL block (green) reads the frozen prefix at every seventh layer (grey); the layers in between (white) are not read.

3.2 REAL-WORLD OFFLINE-TO-ONLINE REINFORCEMENT LEARNING

We first train the actor and critic offline on teleoperation episodes, including failures, and then continue online on the robot's own rollouts with the same objectives.

Compact residual action space. A raw action chunk, with 16 steps of 62-dimensional bimanual actions, spans a large search space for RL. To make it compact, we parameterize each rotation as a 3D axis-angle vector and compress the chunk with a fixed encoder $\mathcal{E}$ into 118 whitened PCA coefficients fitted on the released T-Rex dataset (Niu et al., 2026), which capture correlations across joints and time. The state s includes the cached visuomotor features, the base proposal $\mathbf{A}^{\text{base}}$, and the current tactile and proprioceptive observations. Given s, the actor π_ϕ predicts a residual $\boldsymbol{\delta}$.

Writing the base chunk in rotation-vector coordinates as $\mathbf{x}^{\text{base}}$, we obtain:

$$\boldsymbol{\delta} = \pi_\phi(s), \qquad \mathbf{x}^{\text{react}} = \mathbf{x}^{\text{base}} + D\boldsymbol{\delta}. \tag{1}$$

Here D decodes residuals through inverse whitening and the PCA basis, without adding the action mean. We convert $\mathbf{x}^{\text{react}}$ back to the robot action representation to obtain $\mathbf{A}^{\text{react}}$. The critic evaluates the corrected action code ($\mathcal{E}(\mathbf{A}^{\text{base}})+\boldsymbol{\delta}$) rather than the residual alone, following action-chunk value learning (Li et al., 2025).

Critic learning with milestone rewards. We apply RL throughout complete manipulation trajectories, rather than restricting residual control to a selected phase of the motion. For our long-horizon tasks, which can often exceed 1,000 control frames, this requires credit assignment across multiple stages. In preliminary experiments, episode-terminal rewards alone did not yield a Q-function that supported effective policy improvement, motivating intermediate progress supervision.

Recent robot reward-modeling work similarly emphasizes task progress: RoboReward uses a five-level episodic progress rubric (Lee et al., 2026), while SARM models discrete task stages and within-stage progress (Chen et al., 2026a). Motivated by these progress-based formulations, we use four to six human-defined milestones per task, whose annotations directly supply our RL rewards. Each milestone is rewarded only on its first achievement, with positive weights summing to one. The resulting objective measures discounted task progress without requiring a dense geometric reward for each contact interaction. Both successful and failed episodes provide value supervision through the milestones they achieve; the task-specific milestones and weights are listed in Appendix A.1.

To carry this progress across long intervals, we train both critic heads with a 64-frame SARSA target, which sums the rewards over the next 64 frames and bootstraps from the target critic's value of the recorded action; near episode ends, it falls back to the Monte Carlo return.

Our milestone rewards bound the return-to-go to $[0, 1]$. We discretize this interval into 100 uniform value bins and train each critic head with HL-Gauss cross-entropy (Farebrother et al., 2024). Gaussian target smoothing with $\sigma = 0.0075$ distributes probability mass across neighboring bins, so nearby returns receive similar supervision, preserving their numerical proximity rather than treating bins as unrelated classes.

We regularize the critic's values on unseen actions with a Cal-QL-style penalty (Nakamoto et al., 2023). For each replay state s, the penalty samples Gaussian perturbations of the recorded action in the PCA action space and lowers their values relative to the recorded action. Following Cal-QL, the perturbed values are lower-bounded by the Monte Carlo return of the recorded trajectory from s, minus a small margin, which avoids overly pessimistic values.

Actor learning with behavior regularization. The actor maximizes the critic's value while a behavior regularization keeps its residual close to a reference residual $\boldsymbol{\delta}^{\text{ref}}$. The reference residual is defined as $\boldsymbol{\delta}^{\text{ref}} = \mathcal{E}(\mathbf{A}^{\text{ref}}) - \mathcal{E}(\mathbf{A}^{\text{base}})$ when the recorded action chunk $\mathbf{A}^{\text{ref}}$ reaches a milestone, and $\mathbf{0}$ otherwise. In this way, the actor never imitates failed actions. The actor minimizes

$$\mathcal{L}_\pi = \mathbb{E}_{\mathcal{D}}\left[-\min_{i=1,2} Q_{\psi_i}\big(s, \mathcal{E}(\mathbf{A}^{\text{base}}) + \boldsymbol{\delta}\big) + \lambda_{\text{BC}} \|\boldsymbol{\delta} - \boldsymbol{\delta}^{\text{ref}}\|_2 \right], \tag{2}$$

where $\mathcal{D}$ is the replay data, λ_{BC} weights the behavior regularization, and the minimum over the two Q heads reduces overestimation (Fujimoto et al., 2018). See Appendix A.7 for details and Algorithm 1 for the training procedure.

3.3 Training Recipe

We train T-Rex-RL with a multi-stage recipe that moves from general pretraining data to task-specific data (Appendix A.5).

General stages. The general stages build task-agnostic priors. The latent and base-action experts start from a checkpoint pretrained on more than 20,000 hours of human data and are then supervised on the released robot dataset and our teleoperation data. With these experts frozen, *diverse-task offline RL midtraining* trains the RL experts on milestone-annotated teleoperation data, including failures. The actor and critic blocks are initialized from separate copies of the base-action expert's matched-depth weights, and the actor's final projection is zero-initialized, so training starts from the base behavior.

Table 1: Task-progress scores across policies.

Method	Scissors Cut	Tong Transfer	Swipe Card	Game Controller	Extract Pen Cartridge	Average
ViTacFormer	11	4	6	16	0	7
RDP	14	20	18	55	2	22
$\pi_{0.5}$ (+tac)	23	42	38	69	6	36
T-Rex (Niu et al., 2026)	40	52	41	91	8	46
T-Rex-RL	79	78	64	100	19	68

Table 2: Effect of online learning and model components.

Variant	Scissors Cut	Tong Transfer	Swipe Card	Average
T-Rex	40	52	41	44
incl. online data	44	52	39	45
w/o online RL	43	40	33	39
w/o async	71	71	58	67
w/o tactile	52	49	41	47
T-Rex-RL	79	78	64	74

Task-specific stages. The task-specific stages adapt these priors to each task. We supervise the latent and base-action experts on task-specific teleoperation data, freeze them, and train the RL experts offline on the same data. Online learning uses RLPD (Ball et al., 2023) style 50:50 mix of offline and online data, keeping the same losses. All RL stages use 4 critic updates per actor update, and rollouts limit corrections to 5% of each action dimension's range around the base proposal.

3.4 Real-Time Execution and Online Learning Infrastructure

T-Rex-RL issues high-level action commands at 30 Hz, using real-time chunking (RTC) (Black et al., 2025c) to overlap inference with buffered control. Empirically, RTC enables smooth, continuous rollouts by avoiding inference-induced pauses, reducing the timing mismatch with teleoperation data.

A remote learner updates the two approximately 200M-parameter MoT RL experts alongside rollout collection, outside the robot's control path, using cached frozen backbone features. Replay stores executed actions, and actor checkpoints are activated only between episodes. We achieve critic and actor update-to-data (UTD) ratios of 0.4 and 0.1 optimizer steps per new online control frame; learning typically completes within the rollout-and-human-assisted-reset cycle without additional training-induced waiting. Appendix A.8 details the online learning infrastructure, including replay, feature caching, parallel optimization, checkpoint handoff, and update accounting.

4 Experiments

Our evaluation asks whether offline-to-online RL improves a tactile-reactive policy beyond its offline initialization, how performance evolves with real-world interaction, and which components contribute to that improvement.

4.1 Experimental Setup

Robot platform and teleoperation. We use the bimanual platform of Niu et al. (2026): a Dexmate Vega-1 with two 22-DoF Sharpa Wave hands, head and wrist RGB cameras, and fingertip force/torque and deformation sensing. Human demonstrations and autonomous rollouts share the 30 Hz control timeline described in Section 3.4. The teleoperator receives real-time tactile feed-

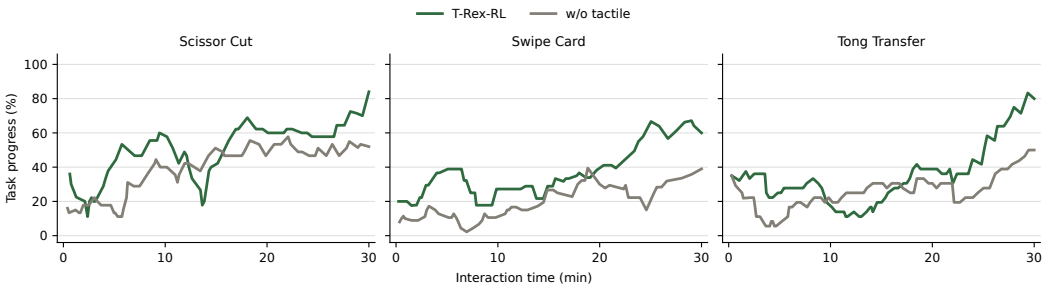


Figure 4: **Online learning progress and tactile ablation.** T-Rex-RL and its no-tactile variant on three tasks. Task-progress scores are averaged over the most recent 7 collection episodes and plotted against cumulative robot execution time. The curves track the changing rollout policies, rather than repeated evaluation of fixed checkpoints. Additional task curves appear in Figure 12.

back during collection, allowing contact information to inform demonstrated actions. Appendix A.2 describes the platform and interface.

Task-specific data and adaptation. For each task, we collect approximately 100 successful teleoperation episodes and retain an additional 20–30 failed attempts. Episodes last about 48 s on average, yielding roughly 1–2 hours of task-specific data. Starting from the diverse-task midtrained weights, we first fine-tune the latent and base-action experts with supervised learning, then freeze them and adapt the RL actor and critic on the task-specific offline data. Online learning subsequently uses approximately 30 minutes of robot execution per task. This budget counts rollout time, not human-assisted resets or total wall-clock training time. Online updates retain the 50/50 offline–online replay mixture.

Initial-state randomization. We vary object positions, orientations, and relative placements within the robot's reachable workspace, using an approximately 30×30 cm placement region. The same task-specific randomization protocol applies to teleoperation, online collection, and evaluation. This tests performance across randomized starts under the training setup, rather than a separate out-of-distribution generalization setting.

Scoring and evaluation protocol. We score each episode by the sum of the weights of its achieved milestones. Weights are positive and sum to one, giving a task-progress score in $[0, 1]$ that credits partial completion; it is not a binary success indicator. Unlike the discounted returns used for critic training, this evaluation score does not discount milestones by when they are reached. Task-specific milestones and weights are provided in Appendix A.1. We evaluate each policy on 16 new randomized episodes per task with a fixed checkpoint, and table entries report the mean score across these episodes. During online collection, we additionally track a moving average over the most recent 7 episodes. This learning curve measures behavior during training, rather than a separate fixed-policy evaluation.

Baselines. We compare with visuo-tactile imitation policies ViTacFormer (Heng et al., 2026) and RDP (Xue et al., 2025), a tactile-conditioned $\pi_{0.5}$ (Black et al., 2025a) baseline, and T-Rex (Niu et al., 2026). These comparisons probe pretrained visuomotor representations and reward-based adaptation in tactile-conditioned policies. The $\pi_{0.5}$ baseline also undergoes midtraining on diverse robot teleoperation data before task-specific adaptation, so the comparison is not simply against a VLA without robot-domain midtraining. Appendix A.3 describes the comparison design.

4.2 Main Results

Table 1 compares task-progress scores between T-Rex-RL and other supervised trained baseline policies. Quantitative results show that T-Rex-RL uniformly outperforms the baselines across all tasks, with an average gain of $+22$ points over the T-Rex supervised learning baseline and $+32$ points over $\pi_{0.5}$ with the same tactile conditioning and 100 hrs of robot teleoperation data midtraining as T-Rex, attesting to the effectiveness of the Mixture-of-Transformers architecture design in

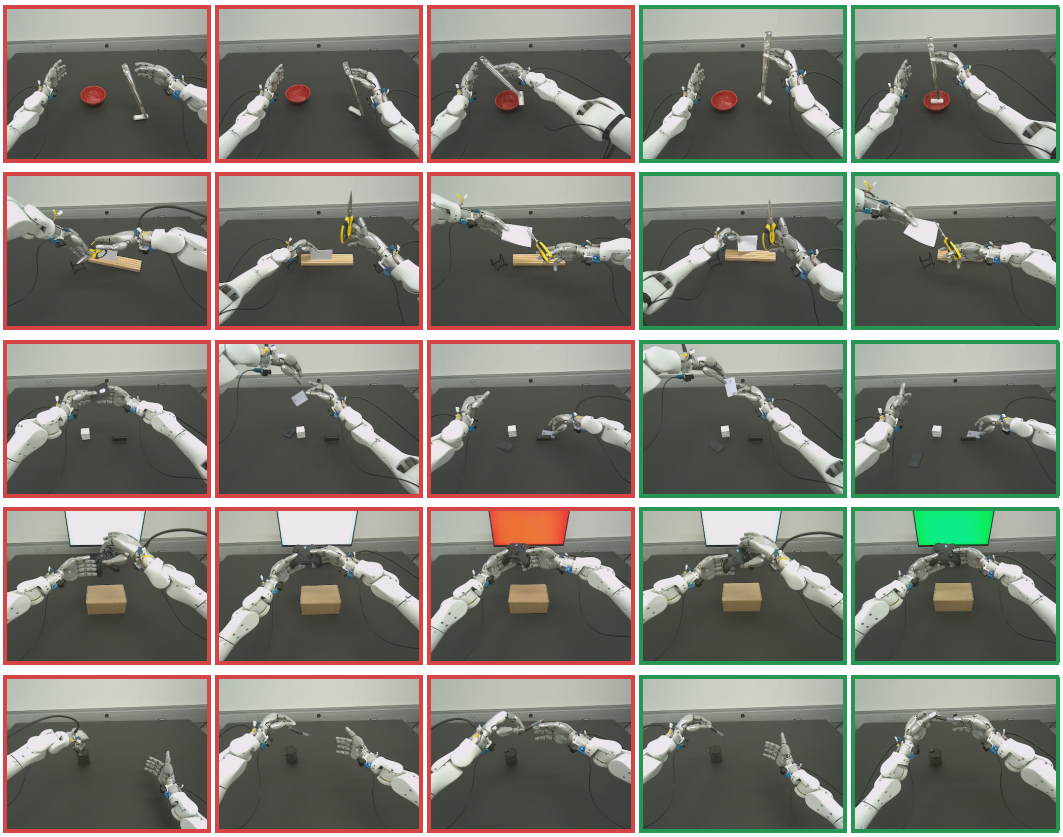


Figure 5: **Qualitative failures and successes.** Red borders mark failures; green borders mark successful task steps. **Row 1: Tong Transfer** dropped tong, failed tong closure, failed transport, tong closed, successful transport. **Row 2: Scissors Cut** imprecise hand placement, failed paper grasp, impecise cut, successful paper grasp, accurate cut. **Row 3: Swipe Card** bimanual hand misplacement, dropped during handover, unaligned swipe, firm handover, precise swipe. **Row 4: Game Controller** dropped controller, imprecise joystick push, imprecise button push, successful handover, accurate button press. **Row 5: Extract Pen Cartridge** imprecise pen grasp, dropped pen, imprecise screw, accurate pen grasp, accurate screw. Rows are ordered top to bottom; examples within each row are ordered left to right.

leveraging multimodal sensory input for dexterous bimanual manipulation. Compared with unpretrained tactile-aware baselines such as ViTacFormer and RDP, T-Rex-RL demonstrates gains of $+61$ and $+46$ points, respectively, highlighting the advantage of pretrained tactile-aware VLAs when diverse robot data are available to effectively exploit their greater model capacity.

During online learning, we observe qualitative improvements in task execution. Figure 5 illustrates selected failures (red borders) and success patterns (green borders) across all five tasks. With only supervised training and offline RL, the policies often exhibit suboptimal behaviors. The robot may grasp the scissors with its left hand too high to prevent a stable grasp or move its right hand too far when reaching for the paper during the Scissors Cut task, grasp the tong too high when attempting to pick it up or move the tong too low when using it to lift the object, or misalign the card in its right hand with the card reader. As online learning progresses, real world rollout experience helps correct these behaviors, leading to more precise and stable task execution over time. In addition, the online reward signals enable the policies to refine their behaviors and improve efficiency: in the Game Controller task the base policy initially attempts to presses the target button multiple times before even reaching the correct position, and with online learning it eventually learns to first move to the correct position before pressing the button. For simplier tasks such as Game Controller, T-Rex-RL continues to improve by becoming robust to randomized object placements. For the hardest Extract

Pen Cartridge task, online learning enables T-Rex-RL to sometimes pick up the pen with the left hand and position it accurately such that the right hand can use its thumb, index, and middle finger to rotate the cap off the pen.

Figure 4 and Figure 12 plots the online learning curves, showing the smoothed task progress score over online learning time. We observe that initially the policy performance may dip due to overconfidence in the offline RL phase and lack of on-policy failure data providing negative feedback signals, but policy performance gradually recovers milestone by milestone, eventually outperforming the base VLA or offline RL performance with a large margin.

4.3 Ablation Study

We further select Scissors Cut, Tong Transfer, and Swipe Card for ablation studies due to their moderate difficulty. Table 2 summarizes the ablation results. We observe that storing the successful online policy rollout segments and appending them to the supervised finetuning dataset to posttrain T-Rex marginally improves performance ($+1$ compared to the base T-Rex policy and -29 compared to the full T-Rex-RL). One reason is that the additional data amount is relatively small compared to the original teleop data, and another reason is that reinforcing policy's own behavior with limited online experience and no negative feedback provides suboptimal supervision.

We further investigate the impact of various components of our real world reinforcement learning framework. The w/o online RL variant evaluates the offline RL policy before any online updates. We observe that even with Calibrated Q-Learning which pushes down Q value estimation outside of data support, the offline RL phase alone struggles to improve policy performance (scoring -35 compared to the full offline-to-online T-Rex-RL) due to the actor exploiting the local overconfidence geometry of the critic. With additional online episodes providing positive and negative feedback, the online phase is able to mitigate overconfidence exploitation and improve the overall task performance.

To ablate the tactile reactive design, we first compare T-Rex-RL with a variant where the RL Actor Expert does not asynchronously refine the Base Action Expert action chunk output at multiple offsets and only refines once per action chunk at offset 0. The no asynchronous refinement variant performs worse by -7, highlighting the importance of frequent tactile feedback in guiding the policy's actions. Finally, we compare with a variant completely removing tactile inputs for the RL Actor and Critic Experts. Figure 4 also includes the online learning curves comparison between the full T-Rex-RL and the no-tactile variant. We observe a -27 drop in performance, underscoring the important role that tactile feedback plays in the reinforcement learning process.

5 Conclusion

We presented T-Rex-RL, an offline-to-online RL framework for improving tactile-reactive VLAs through real-world interaction. T-Rex-RL parameterizes the actor and critic as strided MoT experts, integrating tactile and proprioceptive signals with pretrained visual and action representations. This design gives the RL experts access to intermediate backbone features while limiting the number of trainable layers and preserving the frozen visuomotor prior. The actor asynchronously refines action chunks using fresh tactile feedback, enabling the policy to react to changing contact conditions between base-policy predictions.

Offline RL initializes the actor and critic on teleoperation data, and asynchronous online learning then improves the tactile policy through the robot's own experience. Experiments on bimanual dexterous manipulation show higher task-progress scores than imitation-learning baselines, with ablations highlighting the contributions of tactile feedback, asynchronous action refinement, and online RL. These findings support combining fast tactile reactions with reinforcement learning to improve contact-rich manipulation beyond imitation alone.

The current system still relies on task-specific demonstrations, milestone annotations, and human-assisted resets, and reliable completion of the hardest tasks remains a challenge. Future work could extend this asynchronous tactile RL loop to longer periods of autonomous learning by reducing annotation and reset requirements, and investigate whether learned contact responses transfer across tasks, objects, and robot embodiments.

AI USE STATEMENT

In this work, we used generative AI tools to implement research and engineering ideas and to provide feedback on the interpretation of experimental results and possible research directions. We have not used generative AI tools for proposing or refining hypotheses, cleaning and reformatting datasets, or supporting qualitative data analysis. Generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for refining illustrative figures based on human drawn drafts, refining the paper writing, and assisting literature survey. We have reviewed all AI-assisted work through human review of LLM-generated code, figures, and text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace B. Connors, James Darpinian, Karan Dhabalia, Jared Di Carlo, Danny Driess, Michael Robert 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, Benjamin 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, Alexander 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. In *Proceedings of Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.087. URL https://www.roboticsproceedings.org/rss22/p087.html.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. doi: 10.1177/0278364919887447.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data. In *Proceedings of the International Conference on Machine Learning*, volume 202, pp. 1577–1594. PMLR, 2023. URL https://proceedings.mlr.press/v202/ball23a.html.

Jianxin Bi, Kevin Yuchen Ma, Ce Hao, Mike Zheng Shou, and Harold Soh. VLA-Touch: Enhancing Vision-Language-Action Models with Dual-Level Tactile Feedback. *IEEE Robotics and Automation Letters*, 11(7):8487–8494, 2026. doi: 10.1109/LRA.2026.3692345. URL https://clear-nus.github.io/blog/vla-touch.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In *Proceedings of the Conference on Robot Learning*, volume 305, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *Advances in Neural Information Processing Systems*, volume 38, 2025c. doi: 10.52202/085713-1122. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/300ccb2187dedd4edcc07f7e76d8e553-Abstract-Conference.html.

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. $\pi_{\texttt{rl}}$: Online RL fine-tuning for flow-based vision-language-action models, 2025a. URL https://arxiv.org/abs/2510.25889. arXiv preprint arXiv:2510.25889.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2509.25358.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A Reinforced Fine-Tuning Method for VLA Models via Consistency Policy. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.019. URL https://www.roboticsproceedings.org/rss21/p019.html.

Zhongxi Chen, Yifan Han, Yanming Shao, Huanming Liu, Congsheng Xu, Xiaoyu Chen, Yao Mu, and Wenzhao Lian. BORA: Bridging offline reinforcement learning and online residual adaptation for real-world dexterous VLA models, 2026b. URL https://arxiv.org/abs/2605.30226.

Zhengxue Cheng, Yiqian Zhang, Anni Tang, Keyu Wang, Wenkang Zhang, Haoyu Li, Hengdi Zhang, and Li Song. OmniVTLA: Vision-Tactile-Language-Action Models with Semantic-Aligned Tactile Sensing. *IEEE Robotics and Automation Letters*, 11(9):10146–10153, 2026. doi: 10.1109/LRA.2026.3710060. URL https://doi.org/10.1109/LRA.2026.3710060.

Perry Dong, Kuo-Han Hung, Dorsa Sadigh, and Chelsea Finn. Reinforcement learning for real-time vision-language-action policies. *arXiv preprint arXiv:2609.18207*, 2026. URL https://arxiv.org/abs/2609.18207.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 13049–13071. PMLR, 2024. URL https://proceedings.mlr.press/v235/farebrother24a.html.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396. IEEE, 2017. doi: 10.1109/ICRA.2017.7989385. URL https://is.mpg.de/en/publications/guhollillev17.

Johanna Hansen, Francois Hogan, Dmitriy Rivkin, David Meger, Michael Jenkin, and Gregory Dudek. Visuotactile-RL: Learning multimodal manipulation policies with deep reinforcement learning. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8298–8304, 2022. doi: 10.1109/ICRA46639.2022.9812019.

Liang Heng, Haoran Geng, Kaifeng Zhang, Pieter Abbeel, and Jitendra Malik. ViTacFormer: Learning Cross-Modal Representation for Visuo-Tactile Dexterous Manipulation. In *Proceedings of Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.129. URL https://www.roboticsproceedings.org/rss22/p129.html.

Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-VLA: Unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025.

Jialei Huang, Yang Ye, Yuanqing Gong, Xuezhou Zhu, Yang Gao, and Kaifeng Zhang. Spatially Anchored Tactile Awareness for Robust Dexterous Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2026. URL https://arxiv.org/abs/2510.14647.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021. doi: 10.1177/0278364920987859.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029. IEEE, 2019. doi: 10.1109/ICRA.2019.8794127.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In *Proceedings of the Conference on Robot Learning*, volume 87, pp. 651–673. PMLR, 2018. URL https://proceedings.mlr.press/v87/kalashnikov18a.html.

Dmitry Kalashnikov, Jake Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Scaling Up Multi-Task Robotic Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164, pp. 557–575. PMLR, 2022. URL https://proceedings.mlr.press/v164/kalashnikov22a.html.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. doi: 10.1177/0278364913495721.

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

Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Performant Robotic Manipulation with Real-World Reinforcement Learning. *Science Robotics*, 11(116):eaed6267, 2026. doi: 10.1126/scirobotics.aed6267. URL https://www.science.org/doi/10.1126/scirobotics.aed6267.

Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning Contact-Rich Manipulation Skills with Guided Policy Search. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 156–163. IEEE, 2015. doi: 10.1109/ICRA.2015.7138994. URL https://rll.berkeley.edu/icra2015gps/.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement Learning with Action Chunking. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1856. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/50348e8f9aef984abe0ea1ec2a326f78-Abstract-Conference.html.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-Transformers: A Sparse and Scalable Architecture for Multi-Modal Foundation Models. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2411.04996.

Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning Visuotactile Skills with Two Multifingered Hands. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5637–5643. IEEE, 2025. doi: 10.1109/ICRA55743.2025.11128180. URL https://toruowo.github.io/hato/.

Zixuan Liu, Ruoyi Qiao, Chenrui Tie, Xuanwei Liu, Yunfan Lou, Chongkai Gao, Zhixuan Xu, and Lin Shao. ContactExplorer: Contact coverage-guided exploration for general-purpose dexterous manipulation. *arXiv preprint arXiv:2603.10971*, 2026. URL https://arxiv.org/abs/2603.10971.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16961–16969. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610040.

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. doi: 10.1126/scirobotics.ads5033. URL https://www.science.org/doi/10.1126/scirobotics.ads5033.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets, 2020. URL https://arxiv.org/abs/2006.09359. arXiv preprint arXiv:2006.09359.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated Offline RL Pre-Training for Efficient Online Fine-Tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 62244–62269, 2023. doi: 10.52202/075280-2719. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/c44a04289beaf0a7d968a94066a1d696-Abstract-Conference.html.

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

Gerrit Schoettler, Ashvin Nair, Jianlan Luo, Shikhar Bahl, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5548–5555. IEEE, 2020. doi: 10.1109/IROS45743.2020.9341714.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pp. 387–395. PMLR, 2014. URL https://proceedings.mlr.press/v32/silver14.html.

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In *Proceedings of Robotics: Science and Systems*, 2019. doi: 10.15607/RSS.2019.XV.073. URL https://roboticsproceedings.org/rss15/p73.html.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi "Jim" Fan, Guanya Shi, and Yuke Zhu. Self-Improving Vision-Language-Action Models with Data Generation via Residual RL. In *International Conference on Learning Representations*, 2026. URL https://rpl.cs.utexas.edu/publications/2026/04/01/xiao-iclr26-pld/.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL Token: Bootstrapping Online RL with Vision-Language-Action Models, 2026a. URL https://arxiv.org/abs/2604.23073. arXiv preprint arXiv:2604.23073.

Zhixuan Xu, Yichen Li, Xuanye Wu, Tianyu Qiu, and Lin Shao. FingerEye: Learning dexterous manipulation with continuous vision-tactile sensing. *arXiv preprint arXiv:2604.20689*, 2026b. URL https://arxiv.org/abs/2604.20689.

Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive Diffusion Policy: Slow-Fast Visual-Tactile Policy Learning for Contact-Rich Manipulation. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.052. URL https://www.roboticsproceedings.org/rss21/p052.html.

Patrick Yin, Tyler Westenbroek, Ching-An Cheng, Andrey Kolobov, and Abhishek Gupta. Rapidly Adapting Policies to the Real World via Simulation-Guided Fine-Tuning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e68274fc4f158dbcbd4dddc672f7ee9c-Abstract-Conference.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Zifan Zhao, Siddhant Haldar, Jinda Cui, Lerrel Pinto, and Raunaq Bhirangi. Touch begins where vision ends: Generalizable policies for contact-rich manipulation. *arXiv preprint arXiv:2506.13762*, 2025. URL https://arxiv.org/abs/2506.13762.

Huaihang Zheng, Yi Yang, Kai Ma, Shenglin Xu, Tian Xie, Guozheng Li, Xiangyu Wang, Yiren Ma, Si Liu, Yinian Mao, and Baoxu Liu. TORL-VLA: Tactile guided online reinforcement learning for contact-rich manipulation. *arXiv preprint arXiv:2606.09337*, 2026. URL https://arxiv.org/abs/2606.09337.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loïc Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot Learning with Implicit World Modeling. In *Proceedings of the Conference on Robot Learning*, volume 305, pp. 3952–3971. PMLR, 2025. URL https://proceedings.mlr.press/v305/zheng25a.html.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657. IEEE, 2019. doi: 10.1109/ICRA.2019.8794102.

Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real world robotic reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJe2syrtvS.

A Appendix

A.1 Task Descriptions, Milestones, and Scoring Rubrics

We evaluate T-Rex-RL on five dexterous manipulation tasks using the bimanual platform described in Section 4: Scissors Cut, Tong Transfer, Swipe Card, Game Controller, and Extract Pen Cartridge. For each task, we provide the policy's text instruction, describe the intended behavior, and list the milestones and scoring weights.

Progress on every task is measured with an additive milestone rubric: an ordered set of human-defined milestones whose positive weights sum to one. The rubric serves both training and evaluation. For training, annotators mark the frame at which each milestone is first reached in every teleoperation episode and these labels define the progress reward of Equation equation 9. For evaluation, an episode's score is the undiscounted sum of the weights of the milestones it achieves, so partial completion earns partial credit and a score of one corresponds to completing the task. Below, each task is accompanied by a figure showing the initial scene of a successful teleoperation episode, followed by head-camera frames at the moment each milestone is reached.

Task I: Scissors Cut. *Text Instruction:* "There is a pair of scissors lying on a holder on the left and a blank sheet of paper inserted in a holder on the right; use your left hand to pick up the scissors holding on its center. Lift the scissors and insert your right hand index/middle/ring fingers into the handle of the scissors and expand your right hand's fingers to lock your fingers in; pick up the paper with your left hand holding on its top-left corner; place your right thumb in the other handle of the scissors; finally cut a small piece of paper off from the top-right corner of the blank paper in your left hand."

The robot must pick up a pair of scissors with its left hand, thread three fingers of its right hand into one handle and lock them in place, pick up a sheet of paper with its left hand, place its right thumb in the other handle to open the blades, and close the scissors to cut off a small piece of the paper's top-right corner.

Milestone Rubric (additive):

- $+0.2$: (a) Picks up the scissors from the holder with the left hand.
- $+0.2$: (b) Inserts the right index, middle and ring fingers into one handle and spreads them to lock the grasp.
- $+0.2$: (c) Picks up the paper with the left hand by its top-left corner.
- $+0.2$: (d) Places the right thumb in the other handle and opens the scissors.
- $+0.2$: (e) Cuts a piece off the top-right corner of the paper.

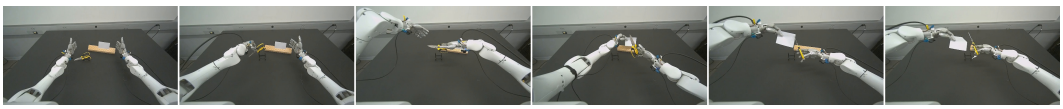


Figure 6: Task I: Scissors Cut. The initial scene, followed by head-camera frames at the moment each milestone (a)–(e) is reached, from left to right.

Task II: Tong Transfer. *Text Instruction:* "There is a metal tong, a coiled small pack of plastic bags, and a red bowl on the table. Use your right hand to pick up the tong, move the tong's end to the pack of plastic bags, close the tong to lift it up, transport it to the red bowl, open the tong to drop the pack of plastic bags in the bowl, and place the tong down on the table."

The robot must pick up a pair of metal tongs with its right hand, grip a small pack of plastic bags by closing the tongs, carry it over a red bowl, open the tongs to drop it inside, and put the tongs back down on the table.

Milestone Rubric (additive):

- $+0.2$: (a) Picks up the tong with the right hand.

- $+0.2$: (b) Closes the tong on the pack of plastic bags and lifts it off the table.
- $+0.2$: (c) Transports the pack above the red bowl.
- $+0.2$: (d) Opens the tong and releases the pack inside the bowl.
- $+0.2$: (e) Places the tong down on the table.

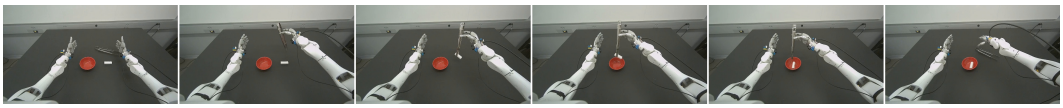


Figure 7: Task II: Tong Transfer. The initial scene, followed by head-camera frames at the moment each milestone (a)–(e) is reached, from left to right.

Task III: Swipe Card. *Text Instruction:* "Use your left hand to pick up the black wallet leaning on the white cube, use your right thumb to slide the card out through the central opening in the black wallet, put down the black wallet using your left hand, adjust your grasp of the card with both hands, and finally use your right hand to swipe the card through the card reader on the table."

The robot must lift a wallet with its left hand, push a card out through the wallet's central opening with its right thumb, set the wallet down, pass the card between its hands to regrasp it, and swipe it through a card reader with its right hand.

Milestone Rubric (additive):

- $+0.2$: (a) Picks up the wallet with the left hand and lifts it off the cube.
- $+0.2$: (b) Slides the card out of the wallet's central opening with the right thumb.
- $+0.2$: (c) Puts the wallet down with the left hand while keeping hold of the card.
- $+0.2$: (d) Adjusts the grasp on the card with both hands, ready to swipe.
- $+0.2$: (e) Swipes the card through the card reader with the right hand.

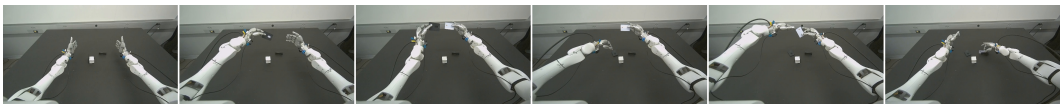


Figure 8: Task III: Swipe Card. The initial scene, followed by head-camera frames at the moment each milestone (a)–(e) is reached, from left to right.

Task IV: Game Controller. *Text Instruction:* "Use your right hand to pick up the black xbox controller from the brown box, hand it to your left hand, use your right thumb to push forward the right joystick until the display turns red, and use your right thumb to press the A button so that the display turns green."

The robot must pick up an Xbox controller from a box with its right hand, hand it over to its left hand, push the right joystick forward with its right thumb until the display turns red, and press the A button with its right thumb to turn the display green.

Milestone Rubric (additive):

- $+0.25$: (a) Picks up the controller from the box with the right hand.
- $+0.25$: (b) Hands the controller over to the left hand.
- $+0.25$: (c) Pushes the right joystick forward with the right thumb; the display turns red.
- $+0.25$: (d) Presses the A button with the right thumb; the display turns green.

Task V: Extract Pen Cartridge. *Text Instruction:* "Pick up the pen in the pen holder with your left hand, hold it horizontally and use your right hand to unscrew the front cap of the pen; once the cap



Figure 9: Task IV: Game Controller. The initial scene, followed by head-camera frames at the moment each milestone (a)–(d) is reached, from left to right.

is off tilt the pen up and use your right hand to take off the front cap and drop it in the pen holder, take out the pen cartridge and drop it the pen holder, and use your left hand to drop the latter part of the pen into the pen holder."

The robot must take a pen out of a pen holder with its left hand, hold it horizontally while its right hand unscrews the front cap, drop the cap into the holder, pull out the ink cartridge and drop it into the holder, and finally drop the rest of the pen into the holder with its left hand.

Milestone Rubric (additive):

- $+0.1$: (a) Picks up the pen from the pen holder with the left hand.
- $+0.1$: (b) Starts unscrewing the front cap with the right hand.
- $+0.25$: (c) Finishes unscrewing the front cap.
- $+0.25$: (d) Drops the front cap into the pen holder.
- $+0.2$: (e) Takes out the cartridge and drops it into the pen holder.
- $+0.1$: (f) Drops the rest of the pen into the pen holder with the left hand.



Figure 10: Task V: Extract Pen Cartridge. The initial scene, followed by head-camera frames at the moment each milestone (a)–(f) is reached, from left to right.

A.2 Teleoperation System Description

Our hardware follows the setup of Niu et al. (2026). Each hand has five instrumented fingertips. A ZED X Mini supplies the head view, and two ZED X One S cameras supply wrist views. Manus gloves capture finger motion, while VIVE trackers capture wrist poses; retargeting maps these signals to robot commands. Figure 11 illustrates the inherited arrangement.

During our data collection, the operator receives tactile feedback in real time. We record both successful and failed attempts, together with the observations and executed actions needed for supervised adaptation and RL. The 30 Hz recording convention matches autonomous rollouts, so both sources can supply action windows to the same replay pipeline. Task-specific milestone annotations provide the progress labels used for value learning and evaluation.

A.3 Baseline and Ablation Protocols

Comparison groups. ViTacFormer and RDP provide visuo-tactile imitation-learning comparisons. We include a single $\pi_{0.5}$ baseline with tactile conditioning and robot-teleoperation midtraining, while T-Rex supplies the closest tactile-reactive supervised comparison. We distinguish these cross-method comparisons from the offline-only variant of T-Rex-RL: the latter retains our architecture and offline RL initialization and removes only subsequent online learning.

Robot-domain adaptation. The tactile-conditioned $\pi_{0.5}$ baseline underwent diverse 100 hours robot teleoperation data midtraining before task-specific adaptation. This controls for the presence

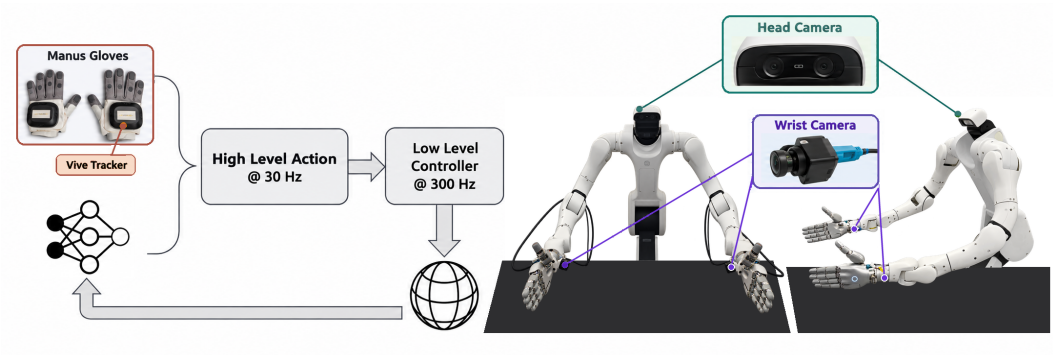


Figure 11: **Robot and teleoperation setup.** The figure shows the head and wrist cameras placements and provides an overview of the robot setup consisting of a Dexmate Vega 1 Bimanual Robot and a pair of mounted Sharpa Wave Dexterous Hands. During teleoperation, the operator uses a pair of Manus Glove Pro Haptic gloves and mounted VIVE Ultimate Trackers to provide realtime end effector and hand pose high level actions. During policy rollout, the policy provide the high level actions. High level actions are expected at 30Hz, which are fed into a low level controller running at 300Hz to drive the robot joints.

of a robot-domain midtraining stage, but does not make the models' original pretraining corpora, parameter counts, or objectives identical. Matching downstream experience therefore does not eliminate differences in upstream initialization. The final evaluation uses the common task rubric and initial-state randomization defined in Section 4.

A.3.1 Ablation Design

Tactile observations and expert depth. The no-tactile comparison removes contact observations from both RL streams while retaining visual context, proprioception, and the base proposal. This tests whether contact-conditioned policy improvement contributes beyond residual RL alone. We also attempted to replace the four-block strided RL experts with shallow actor and critic heads, but could not obtain consistently stable offline RL training: these runs frequently diverged before producing a policy suitable for online evaluation. Although this failed variant does not provide a controlled quantitative comparison, its optimization difficulty is consistent with the challenge of jointly fusing high-dimensional visual, proprioceptive, tactile, and action representations in shallow networks. This observation motivates giving the RL experts repeated access to pretrained intermediate features through the strided MoT design, rather than requiring shallow heads to learn this multimodal fusion on their own. Because reducing depth changes both access to intermediate backbone features and trainable capacity, these failed runs do not isolate either factor and are reported only as an optimization finding.

Online experience. The offline-only policy is evaluated before any online updates, with the same task-adapted backbone and scoring protocol as the final policy. This measures the benefit of the complete online phase, including its additional experience and optimization; it is not a compute-matched test of data source alone.

A.4 Additional Online Learning Curves

Figure 12 complements the three-task tactile ablation in Figure 4 with online learning curves for game-controller manipulation and pen-cartridge extraction. The plotting protocol is the same: task-progress scores averaged over the most recent 7 collection episodes, against cumulative robot execution time.

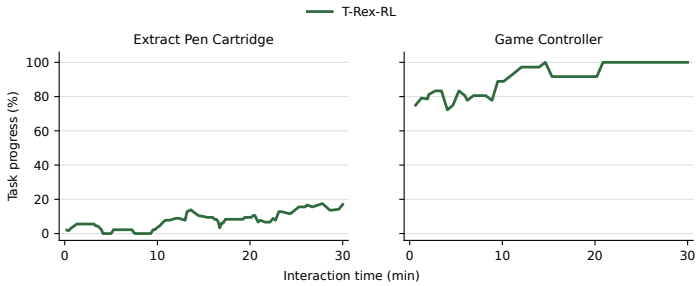


Figure 12: **Online learning progress on additional tasks.** Game-controller manipulation and pen-cartridge extraction, using the same collection-time scoring protocol as Figure 4.

A.5 Training Recipe and Data Sources

Human-data pretraining. We initialize the latent and base-action experts from the pretrained T-Rex checkpoint (Niu et al., 2026), which uses more than 20,000 hours of diverse human data. We reuse this initialization rather than repeating large-scale pretraining. It supplies visual-language representations and an action-generation prior for subsequent robot learning.

Visuomotor midtraining. We continue supervised training of the latent and base-action experts on 100 hours of teleoperation data: 50 hours from the publicly released T-Rex robot dataset, available on Hugging Face, and 50 hours collected in-house. Following the inherited visuomotor training procedure, action flow matching trains the base proposals and future-latent alignment trains predictive scene representations (Equations equation 4 and equation 3). This stage adapts the human-pretrained policy to diverse robot manipulation behaviors before introducing reward-based optimization.

Diverse-task offline RL midtraining. We annotate a subset of the collected diverse-task teleoperation trajectories with milestone labels and include collected failure trajectories in the offline RL data. We initialize the RL actor and critic from separate copies of matched-depth base-action weights: their 4 Transformer blocks inherit layers $\{7, 14, 21, 28\}$ of the midtrained base-action expert. The actor's final output projection is zero-initialized so that residual learning starts from the base proposal. We then freeze the latent and base-action experts and train the RL actor and critic with the objectives in Section 3.2. Milestone-reaching segments supply behavior anchors, while both successful and failed trajectories contribute to value learning. This stage provides a shared actor–critic initialization across tasks before downstream specialization.

Task-specific supervised and offline RL adaptation. For each downstream task, we collect human teleoperation trajectories containing both successes and failures. We first perform supervised fine-tuning of the latent and base-action experts on task-specific teleoperation data. We then freeze the adapted visuomotor experts and continue training the previously midtrained RL actor and critic on the task's offline data. Base proposals and residual behavior targets are obtained using the task-adapted supervised policy. This ordering separates adaptation of the underlying visuomotor behavior from reward-based learning of tactile corrections; it also ensures that the task-specific critic learns against the base policy used for subsequent rollouts.

Online improvement. We deploy the task-adapted base policy together with the residual actor, add executed robot trajectories to online replay, and continue training the RL experts while keeping the visuomotor experts frozen. The objectives remain unchanged; online learning uses the replay mixture in Equation equation 16. All RL stages, including diverse-task midtraining, use 4 critic optimization steps per actor optimization step. Task-specific data volumes are reported in the experiments, and implementation details appear in Appendix A.7.

A.6 Architecture and Supervised Objectives

This section expands the four expert streams summarized in Section 3.1. We write the non-tactile input as $\mathbf{obs}_t = \{\ell, \mathbf{im}_t, \mathbf{p}_t\}$ and the tactile input as $\mathbf{tac}_t = \{\mathbf{f}_{t-H_f:t}, \mathbf{d}_t\}$, where ℓ is the instruction,

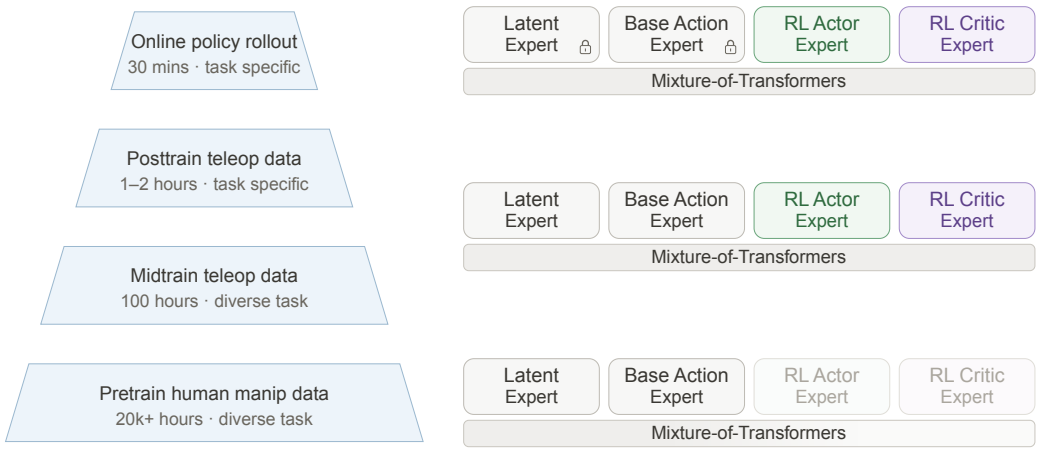


Figure 13: **Data pyramid and expert use.** The central expert row summarizes supervised-then-offline-RL training for both teleoperation tiers. Faded regions are inactive; locks denote frozen experts. Tier widths are schematic.

$\mathbf{im}_t$ contains head and wrist images, $\mathbf{p}_t$ is proprioception, and $\mathbf{f}, \mathbf{d}$ denote fingertip force/torque measurements and deformation maps. The proposal horizon is $H = 16$ control frames.

Latent expert and future visual prediction. The latent expert uses a 28-layer Qwen3VL-2B backbone to encode the head-camera image and language instruction. To make its representations informative about how an interaction will evolve, we use future latent representation alignment (FLARE) (Zheng et al., 2025) throughout visuomotor training. Thirty-two learned queries predict features $\widehat{\mathbf{z}}_{k,m}$ for 8 future frames k, with 4 tokens m per frame. Given target features $\mathbf{z}_{k,m}$ from the same frozen vision encoder used for current observations, we minimize their mean cosine distance,

$$\mathcal{L}_{\text{future}} = \mathbb{E}\left[\frac{1}{32}\sum_{k=1}^{8}\sum_{m=1}^{4}\left(1 - \frac{\widehat{\mathbf{z}}_{k,m}^{\top}\mathbf{z}_{k,m}}{\|\widehat{\mathbf{z}}_{k,m}\|_2\|\mathbf{z}_{k,m}\|_2}\right)\right], \tag{3}$$

with the targets detached from the computation graph. This objective matches embedding directions without constraining their magnitudes or decoding pixels. Its gradients train the latent stream, learned queries, and prediction head against a fixed visual representation. Future frames are used only to construct supervision targets; the attention mask isolates the future-prediction branch from the action-generating streams. The queries are omitted during RL and deployment, when the latent expert is frozen. Appendix A.6.1 provides the token layout and training details.

Base-action expert and flow matching. The base-action expert has 28 Transformer layers and conditions on the latent context, wrist-camera features, a noisy action chunk $\mathbf{x}_\tau$, and a flow timestep τ. It learns a vector field v_θ that transforms Gaussian noise into an action proposal. For a demonstrated chunk $\mathbf{A}^{\text{demo}}$, training samples $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\tau \sim \text{Beta}(1.5, 1.0)$, and constructs

$$\mathbf{x}_\tau = (1-\tau)\mathbf{A}^{\text{demo}} + \tau\boldsymbol{\epsilon}, \qquad \mathcal{L}_{\text{FM}} = \mathbb{E}\left[\left\|v_\theta(\mathbf{x}_\tau, \tau; \mathbf{c}_t^{\text{vis}}) - (\boldsymbol{\epsilon} - \mathbf{A}^{\text{demo}})\right\|_2^2\right], \tag{4}$$

where $\mathbf{c}_t^{\text{vis}}$ denotes the visual-language conditioning available to the base-action expert. The two streams are jointly supervised by $\mathcal{L}_{\text{FM}} + \lambda_{\text{future}}\mathcal{L}_{\text{future}}$, with $\lambda_{\text{future}} = 0.5$: future-feature alignment trains temporally informative representations, while flow matching trains action generation. These objectives support pretraining on retargeted human motions followed by adaptation to robot demonstrations. At inference, we initialize $\mathbf{x}_1$ with Gaussian noise and numerically integrate $d\mathbf{x}_\tau/d\tau = v_\theta$ from $\tau = 1$ to 0 using 10 Euler steps, obtaining $\mathbf{A}_{t:t+H}^{\text{base}} = \mathbf{x}_0$. Thus, the base expert produces a complete action proposal, and tactile adaptation operates through a learned residual on that proposal.

Cached visual-action context. During RL, the latent and base-action experts are frozen. We retain their layerwise keys and values as a shared prefix,

$$\mathbf{K}_t = \text{sg}\Big(\Big[\mathbf{KV}_t^{\text{lat}} \mid \mathbf{KV}_t^{\text{base}}\Big]\Big), \tag{5}$$

where sg denotes stop-gradient. Thus, each actor refinement or critic computation can reuse the VLA's hierarchical representations without rerunning its visual backbone or perturbing its pretrained parameters.

Tactile tokens. We encode each fingertip through three complementary branches. For temporal contact dynamics, a vector-quantized autoencoder (VQ-VAE) maps a 16-frame history ($H_f = 15$) of six-dimensional force/torque measurements to one discrete code. A temporal convolutional encoder with two strided blocks and mean pooling produces a 256-dimensional embedding, which is assigned to its nearest entry in a 64-code vocabulary. Convolutional weights are shared across fingers, with learned finger identities distinguishing their inputs. The tokenizer is pretrained to reconstruct force histories using magnitude-weighted squared error so that high-force contacts contribute more strongly; its codebook is updated by exponential moving averages. Separately, a direct projection of the current force vector preserves instantaneous information that temporal pooling and quantization may suppress.

For spatial contact structure, each fingertip supplies a single-channel deformation map describing local skin displacement. A compact ResNet-18 encoder retains three residual stages, each followed by a 3×3 convolution with 128 output channels. Its spatial features are flattened and projected into tactile tokens. The deformation encoder is pretrained by autoencoding and kept frozen during policy learning. Concatenating the temporal codes, current-force features, and deformation features across fingers gives

$$\mathbf{z}_t^{\mathrm{tac}} = \left[\mathrm{Emb}_{\mathrm{vq}}\big(E_f(\mathbf{f}_{t-H_f:t})\big)\,;\,\mathrm{Proj}_f(\mathbf{f}_t)\,;\,\mathrm{Proj}_d(E_d(\mathbf{d}_t))\right]. \tag{6}$$

This representation exposes force dynamics, instantaneous loading, and spatial contact geometry to both RL experts. Tactile tokens are refreshed at each actor call even while the visual-action prefix remains cached.

Residual actor expert. At each intra-chunk offset $t_o \in \{0, 4, 8, 12\}$, the actor receives the latest tactile tokens, proprioception, the base proposal, and an action-query token. Through masked attention, it additionally reads the cached prefix $\mathbf{K}_t$ and predicts a residual action chunk

$$\begin{aligned} \Delta\mathbf{A}_{t:t+H} &= \pi_\phi\big(\mathbf{z}^{\mathrm{tac}}_{t+t_o}, \mathbf{p}_{t+t_o}, \mathbf{A}^{\mathrm{base}}_{t:t+H}; \mathbf{K}_t\big)\,, \\ \mathbf{A}^{\mathrm{react}}_{t:t+H} &= \mathbf{A}^{\mathrm{base}}_{t:t+H} + \Delta\mathbf{A}_{t:t+H}. \end{aligned} \tag{7}$$

The output retains the original chunk's time coordinates at every refinement call. For a chunk starting at $t = 0$, the actor at offset $t_o = 4$ uses the original visual-action cache and fresh tactile observations, but still corrects the proposal for frames $0{:}16$, rather than producing a new chunk for $4{:}20$. Only uncommitted, currently scheduled actions can affect execution. The base chunk anchors the policy to the pretrained behavior, while the residual focuses RL capacity on contact-dependent corrections that are difficult to capture from demonstrations alone.

Twin critic expert. The critic receives the same cached context, tactile tokens, and proprioception as the actor, together with the candidate action chunk to be evaluated. Two learned Q-query tokens and separate output heads produce $Q_{\psi_1}(s_{t+t_o}, \mathbf{A}_{t:t+H})$ and $Q_{\psi_2}(s_{t+t_o}, \mathbf{A}_{t:t+H})$, where s_{t+t_o} comprises the cached context and current tactile and proprioceptive observations. Each head predicts probabilities over 100 value bins; its scalar Q-value is the probability-weighted mean of the bin centers (Appendix A.7). The two estimates share the critic's Transformer stream. Conditioning value prediction on contact information lets the critic distinguish visually similar states with different force or deformation patterns, while sharing the frozen prefix avoids duplicating the visual backbone.

Strided RL experts. The actor and critic need access to the backbone's intermediate representations, but a full-depth, 28-layer expert contains 1.41B parameters, making separate full-depth RL streams costly to update repeatedly. We therefore use a stride of 7 across backbone depth: each RL expert contains 4 attention-and-feed-forward blocks, aligned with prefix layers $\{7, 14, 21, 28\}$ (Figure 3). This reduces the parameter count to approximately 200M per expert, about 7 times smaller than the full-depth counterpart. Block r reads the cached keys and values at prefix layer $7r$ and updates its own stream's tokens. These updated tokens pass directly to the next RL block; the intervening backbone layers have no corresponding actor or critic blocks. The stride refers to Transformer depth and is separate from the 4-control-step interval between tactile updates.

The attention mask permits each expert to read the frozen prefix and its own permitted token positions, but prevents the prefix from reading RL tokens and prevents direct attention between the

actor and critic streams. Critic fitting therefore leaves the pretrained policy and actor parameters unchanged; actor improvement can still use the critic's derivative with respect to the candidate action. The critic is omitted at deployment.

A.6.1 FUTURE LATENT ALIGNMENT

The learned query tensor has shape $1 \times 32 \times d_{\text{hid}}$, where d_{hid} is the latent expert's hidden dimension, and is broadcast across the batch. The queries are appended after the L_{slow} visual-language tokens, with multimodal rotary positional embeddings (M-RoPE) continuing beyond the last observation token. After the latent expert's forward pass, we extract the 32 query states at positions L_{slow} through $L_{\text{slow}} + 31$ and pass them through a two-layer prediction MLP with GELU. The implementation permits the prediction head to read a selected hidden layer through `flare_layer_index`; an index of -1 selects the final normalized states, while earlier layers can supply intermediate predictive representations.

We sample future frames at offsets $4, 8, \ldots, 32$; at 30 Hz, these span approximately 0.13–1.07 s. Adaptive average pooling reduces each frame's vision-encoder patch sequence to 4 embeddings, and concatenating the 8 groups gives a $B \times 32 \times d_{\text{vis}}$ target tensor, where d_{vis} is the visual embedding dimension and the prediction head maps to this same dimension. Query and target tokens are aligned one-to-one in temporal order. This prediction window extends beyond the 16-step action chunk, providing supervision about subsequent scene evolution.

In the inherited cascaded policy's supervised training, the alignment term accompanies both the base-action and tactile flow losses; it is distinct from the subsequent RL objectives. The implementation logs it separately as `flare_loss`. When a required future frame is unavailable near an episode boundary, the batch omits the alignment term and retains the action-training losses.

A.7 REINFORCEMENT LEARNING OBJECTIVES AND IMPLEMENTATION

We provide the complete objectives behind Section 3.2. All discounts and episode indices refer to individual 30 Hz control frames.

State and action representation. At control frame u, the RL state s_u comprises the cached visual-action context, the base proposal, and current tactile and proprioceptive observations. The proposal's time coordinates follow the convention in Appendix A.6.

Each bimanual control has 62 coordinates: per side, 22 relative hand-joint coordinates, 3 relative translations, and a six-dimensional relative-rotation representation. All steps in a chunk are expressed relative to its starting state. Before PCA, we convert each rotation to a three-dimensional rotation vector, reducing the chunk from $16 \times 62 = 992$ to $16 \times 56 = 896$ coordinates. We fit PCA on full action chunks from the diverse released dataset (Niu et al., 2026), not on residuals, and retain 118 whitened coefficients. This fixed representation captures correlations across joints and time while reducing the dimension of policy optimization.

Let $\mathcal{E}(\mathbf{A})$ denote this action encoding and $\mathbf{a}_u^{\text{base}} = \mathcal{E}(\mathbf{A}_u^{\text{base}})$. The actor π_ϕ predicts a residual $\pi_\phi(s_u) \in \mathbb{R}^{118}$:

$$\mathbf{a}_u = \mathbf{a}_u^{\text{base}} + \pi_\phi(s_u). \tag{8}$$

The correction is mapped back through the PCA basis and inverse whitening without adding the action mean: it is a residual on the base proposal, not a separately decoded absolute action. The critic evaluates the full corrected action code, not the residual alone, using two heads $Q_{\psi_i}(s, \mathbf{a})$, $i \in \{1, 2\}$. Conditioning value on a sequence rather than a single control follows the action-chunk perspective of Li et al. (2025).

Milestone-based progress supervision. We train on complete episodes, including long intervals in which no correction is needed. Terminal-only rewards would require credit assignment across more than 1,000 control frames and many coordinated joints. We therefore define human-specified milestones with positive weights w_m summing to one. If $c_{m,u}$ indicates whether milestone m has been achieved by frame u, the reward is

$$r_u = \sum_{m=1}^{M} w_m (c_{m,u+1} - c_{m,u}), \qquad \sum_m w_m = 1. \tag{9}$$

The achievement flags are cumulative, so each milestone is rewarded once. The policy maximizes expected discounted progress, $\mathbb{E}_{\pi_\phi}[\sum_u \gamma^u r_u]$, with discounting applied per control frame. This intermediate supervision addresses sparse feedback without requiring a dense geometric reward; it defines a progress objective rather than guaranteeing preservation of the terminal-only optimum. Task-specific milestones and weights appear in Appendix A.1.

Long-horizon value learning. Replay contains executed controls, including the effects of asynchronous chunking and tactile refinement. Training windows can start at any valid control frame, not only at policy-call boundaries (Section 3.4). Denote a recorded 16-step window by $\mathbf{A}_u^{\mathcal{D}}$ and its code by $\mathbf{a}_u^{\mathcal{D}}$. To propagate progress signals over long episodes, we use a 64-frame SARSA target with per-frame discount $\gamma = 0.999$:

$$y_u = \begin{cases} \displaystyle\sum_{j=0}^{63} \gamma^j r_{u+j} + \gamma^{64} \min_i Q_{\bar{\psi}_i}(s_{u+64}, \mathbf{a}_{u+64}^{\mathcal{D}}), & u+64 < T, \\ G_u^{\mathrm{MC}}, & u+64 \geq T, \end{cases} \qquad G_u^{\mathrm{MC}} = \sum_{j=0}^{T-u-1} \gamma^j r_{u+j}. \tag{10}$$

Here T is the episode length and $\bar{\psi}$ denotes Polyak-averaged target parameters. The minimum reduces value overestimation (Fujimoto et al., 2018). Returns never cross episode boundaries, and all episode endings, including time limits and interruptions, use the same remaining-return convention. The 64-frame target spans multiple policy updates; it does not imply executing one 16-step proposal open loop for 64 frames. Since its endpoint action comes from replay, the target follows recorded behavior rather than providing an unbiased evaluation of the current actor. We accept this approximation to obtain longer-horizon supervision from scarce real-world trajectories.

Binned value prediction with HL-Gauss. Because milestone rewards are nonnegative, awarded only once, and sum to at most one per episode, the discounted return-to-go lies in $[0, 1]$. We divide this interval into 100 equal-width bins with edges $b_k = k/100$, $k = 0, \ldots, 100$, and centers $z_k = (b_{k-1} + b_k)/2$. Each critic head outputs softmax probabilities $p_{\psi_i,k}(s, \mathbf{a})$, with scalar readout

$$Q_{\psi_i}(s, \mathbf{a}) = \sum_{k=1}^{100} z_k \, p_{\psi_i,k}(s, \mathbf{a}). \tag{11}$$

Following HL-Gauss (Farebrother et al., 2024), we encode the scalar target y_u as the bin masses of a Gaussian centered at y_u, normalized over $[0, 1]$:

$$q_k(y_u) = \frac{\Phi((b_k - y_u)/\sigma) - \Phi((b_{k-1} - y_u)/\sigma)}{\Phi((1 - y_u)/\sigma) - \Phi(-y_u/\sigma)}, \qquad \sigma = 0.0075, \tag{12}$$

where Φ is the standard normal cumulative distribution function. Both heads minimize cross-entropy against this detached target:

$$\mathcal{L}_{\mathrm{TD}} = -\mathbb{E}_{\mathcal{D}}\left[\frac{1}{2}\sum_{i=1}^{2}\sum_{k=1}^{100} \mathrm{sg}(q_k(y_u)) \log p_{\psi_i,k}(s_u, \mathbf{a}_u^{\mathcal{D}})\right]. \tag{13}$$

The smoothing width is 0.75 times the bin width, giving nearby scalar targets overlapping supervision rather than an abrupt change between one-hot labels. Bootstrapping, conservative regularization, and actor improvement all use the scalar expectation in Equation equation 11.

Calibrated conservative critics. To discourage the actor from exploiting unsupported high values, we regularize the critic with a Cal-QL-style objective (Nakamoto et al., 2023). For each replay state, we draw 10 negative action codes by adding isotropic Gaussian noise to $\mathbf{a}_u^{\mathcal{D}}$. Thus negatives are centered on recorded human or robot behavior, rather than on an unexecuted base proposal. Their calibrated values and conservative penalty are

$$\begin{aligned} \widehat{Q}_{i,j} &= \max\left\{ Q_{\psi_i}(s_u, \widetilde{\mathbf{a}}_u^{(j)}), \mathrm{sg}(G_u^{\mathrm{MC}}) - \varepsilon_{\mathrm{cal}} \right\}, \\ \mathcal{L}_{\mathrm{cal}} &= \mathbb{E}_{\mathcal{D}}\left[\sum_{i=1}^{2}\left(\log\left(\sum_{j=1}^{10}\exp\left(\widehat{Q}_{i,j}\right)\right) - Q_{\psi_i}(s_u, \mathbf{a}_u^{\mathcal{D}})\right)\right], \end{aligned} \tag{14}$$

The floor uses the observed return minus a small margin $\varepsilon_{\mathrm{cal}} > 0$, allowing local action differentiation before calibration stops further downward pressure on a negative's value. It calibrates the

penalty, not a hard constraint on every Q prediction. We optimize $\mathcal{L}_Q = \mathcal{L}_{\mathrm{TD}} + \beta_{\mathrm{cal}}\mathcal{L}_{\mathrm{cal}}$. Noisy candidates are used only for critic training, never as robot exploration commands.

Behavior-anchored residual improvement. The actor maximizes the smaller critic value while remaining close to a behavior reference. For segments that reach a milestone, the reference is the recorded residual; elsewhere it is zero:

$$\boldsymbol{\delta}_u^{\mathrm{ref}} = \begin{cases} \mathbf{a}_u^{\mathcal{D}} - \mathbf{a}_u^{\mathrm{base}}, & \text{milestone-reaching segment,} \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{15}$$

For human data, we infer the corresponding base proposal with the supervised policy. The actor optimizes Equation equation 2 in the main text. The value gradient passes through the critic's action input into the actor, with critic parameters fixed (Silver et al., 2014). The behavior term is measured in whitened PCA coordinates. Its zero reference encourages leaving the base action unchanged, not issuing a zero robot command. Failed trajectories still train the critic without supplying imitation targets. Together, behavior anchoring and conservative values constrain policy improvement around supported corrections.

Offline-to-online improvement. We first train on offline teleoperation trajectories, then collect robot rollouts and continue learning with equal numbers of offline and online samples in every minibatch (Ball et al., 2023):

$$\mathcal{D}_{\mathrm{mix}} = \tfrac{1}{2}\mathcal{D}_{\mathrm{off}} + \tfrac{1}{2}\mathcal{D}_{\mathrm{on}}. \tag{16}$$

This preserves demonstration coverage while incorporating the contact states, successes, and failures encountered by the deployed policy. We add neither an exploration-noise process nor an entropy bonus to the residual actor. At rollout only, corrections are projected to within 5% of each action dimension's range around the base proposal, and empirically less than 0.5% of the residual corrections from the RL actor exceeds this cap.

A.7.1 Numerical and Deployment Details

Target-network updates. The target critic is updated by Polyak averaging,

$$\bar{\psi} \leftarrow (1-\rho)\bar{\psi} + \rho\psi, \tag{17}$$

where $\rho = 0.005$ is the weight on the current critic, with the remaining 0.995 retained from the previous target parameters.

Regularization parameters. The negative-sample noise scale σ_{neg}, conservative weight β_{cal}, behavior weight λ_{BC}, and calibration margin $\varepsilon_{\mathrm{cal}}$ control the regularization strength.

Deployment constraint. In training and validation checks, more than 99% of human–base action-chunk differences stay within the threshold at every coordinate and step, and fewer than 1% of trained-actor chunks exceed it anywhere. These are whole-chunk fractions, not per-coordinate frequencies. They indicate that the projection is rarely active, but do not by themselves establish that removing it would leave performance unchanged.

A.7.2 Offline-to-Online Actor–Critic Algorithm

Algorithm 1 summarizes task-specific offline adaptation and subsequent online learning, initialized according to Appendix A.5. Both phases use the same update routine. We write $\langle\cdot\rangle_{\mathcal{B}}$ for a minibatch mean and sg for stop-gradient. Here $\mathrm{CE}(\mathbf{q}, \mathbf{p})$ is cross-entropy between the HL-Gauss target in Equation equation 12 and the predicted bin probabilities. Gradient steps below specify the objectives; optimizer settings are separate hyperparameters.

The online loop groups work by newly ingested episodes for clarity; rollout collection and learner updates proceed independently. The counter F counts cumulative newly ingested online control frames. All counters exclude offline learning. Complete update groups give $U_\pi = \lfloor 0.1F \rfloor$ and $U_Q = 4U_\pi$. Appendix A.8 describes frame accounting and checkpoint handoff.

Algorithm 1 T-Rex-RL offline-to-online residual actor–critic learning

Input: Frozen latent/base-action experts; fixed PCA encoder $\mathcal{E}$;
initial residual actor ϕ and twin-head critic ψ; labeled offline replay $\mathcal{D}_{\text{off}}$.
Settings: $H = 16$, $n = 64$, $\gamma = 0.999$, $K = 10$ negatives, even batch size B;
offline update groups N_{off}, rates η_Q, η_π, Polyak weight $\rho = 0.005$,
regularizers $\beta_{\text{cal}}, \lambda_{\text{BC}}, \varepsilon_{\text{cal}}$, noise scale σ_{neg}.
Output: Learned residual actor ϕ and critic ψ; actor deployed with frozen backbone.

Prepare offline rewards $r_u = \sum_m w_m(c_{m,u+1} - c_{m,u})$ and returns $G_u = \sum_{j=0}^{T-u-1} \gamma^j r_{u+j}$; infer base proposals and cache frozen features.
Initialize target critic $\bar{\psi} \leftarrow \psi$.
Phase I: offline learning
for $g = 1, \ldots, N_{\text{off}}$ **do**
  for $k = 1, \ldots, 4$ **do**
    Sample $\mathcal{B}$ of B valid replay windows from $\mathcal{D}_{\text{off}}$.
    UPDATE($\mathcal{B},\ k = 4$)
  end for
end for
Phase II: asynchronous online collection and learning
Initialize $\mathcal{D}_{\text{on}} \leftarrow \emptyset$, $F \leftarrow 0$, $U_Q \leftarrow 0$, $U_\pi \leftarrow 0$; publish ϕ.
while online interaction budget remains or collected episodes await processing **do**
  Ingest new labeled robot episodes into $\mathcal{D}_{\text{on}}$; update F, rewards, returns, and caches.
  Rollouts use RTC, decoded residuals, and the 5%-range projection; record executed controls.
  while $U_\pi < \lfloor 0.1F \rfloor$ **do**
    for $k = 1, \ldots, 4$ **do**
      Sample $B/2$ windows from each of $\mathcal{D}_{\text{off}}$ and $\mathcal{D}_{\text{on}}$ to form $\mathcal{B}$.
      UPDATE($\mathcal{B},\ k = 4$); $U_Q \leftarrow U_Q + 1$.
    end for
    $U_\pi \leftarrow U_\pi + 1$.
  end while
  Publish updated actor ϕ for activation at a subsequent episode boundary.
end while
Shared critic/actor update; all backbone features and replay targets are detached
function UPDATE($\mathcal{B}$, updateActor)
  For each window at u, set $\mathbf{a}_u^{\mathcal{D}} = \mathcal{E}(\mathbf{A}_u^{\mathcal{D}})$, $\mathbf{a}_u^{\text{base}} = \mathcal{E}(\mathbf{A}_u^{\text{base}})$ and

$$y_u = \begin{cases} \sum_{j=0}^{n-1} \gamma^j r_{u+j} + \gamma^n \min_i Q_{\bar{\psi}_i}(s_{u+n}, \mathbf{a}_{u+n}^{\mathcal{D}}), & u+n < T, \\ G_u, & u+n \geq T. \end{cases}$$

  Draw $\boldsymbol{\xi}_j \sim \mathcal{N}(\mathbf{0}, I)$; set $\widetilde{\mathbf{a}}_u^{(j)} = \mathbf{a}_u^{\mathcal{D}} + \sigma_{\text{neg}}\boldsymbol{\xi}_j$, $j = 1, \ldots, K$.
  $\widehat{Q}_{i,j} \leftarrow \max\{Q_{\psi_i}(s_u, \widetilde{\mathbf{a}}_u^{(j)}), \text{sg}(G_u) - \varepsilon_{\text{cal}}\}$.

$$\mathcal{L}_Q \leftarrow \left\langle \frac{1}{2}\sum_{i=1}^{2} \text{CE}\Big(\text{sg}(\mathbf{q}(y_u)), \mathbf{p}_{\psi_i}(s_u, \mathbf{a}_u^{\mathcal{D}})\Big) \right\rangle_{\mathcal{B}} + \beta_{\text{cal}} \left\langle \sum_{i=1}^{2} \left(\log \sum_{j=1}^{K} e^{\widehat{Q}_{i,j}} - Q_{\psi_i}(s_u, \mathbf{a}_u^{\mathcal{D}}) \right) \right\rangle_{\mathcal{B}}.$$

  $\psi \leftarrow \psi - \eta_Q \nabla_\psi \mathcal{L}_Q$; $\bar{\psi} \leftarrow (1-\rho)\bar{\psi} + \rho\psi$.
  if updateActor **then**
    $\boldsymbol{\delta}_u^{\text{ref}} \leftarrow \mathbf{a}_u^{\mathcal{D}} - \mathbf{a}_u^{\text{base}}$ on milestone-reaching segments; $\mathbf{0}$ otherwise.

$$\mathcal{L}_\pi \leftarrow \left\langle -\min_i Q_{\psi_i}(s_u, \mathbf{a}_u^{\text{base}} + \pi_\phi(s_u)) + \lambda_{\text{BC}} \|\pi_\phi(s_u) - \text{sg}(\boldsymbol{\delta}_u^{\text{ref}})\|_2 \right\rangle_{\mathcal{B}}.$$

    $\phi \leftarrow \phi - \eta_\pi \nabla_\phi \mathcal{L}_\pi$, differentiating through the action input with ψ fixed.
  end if
end function

A.8 Real-Time Online Learning System

The system has three independently progressing components: a robot controller and policy server, a durable episode and checkpoint store, and a remote learner (Figure 14). Learning and data preparation run outside the robot's per-step control path. Synchronization occurs through episode records and versioned checkpoints rather than shared access to the robot's local filesystem.

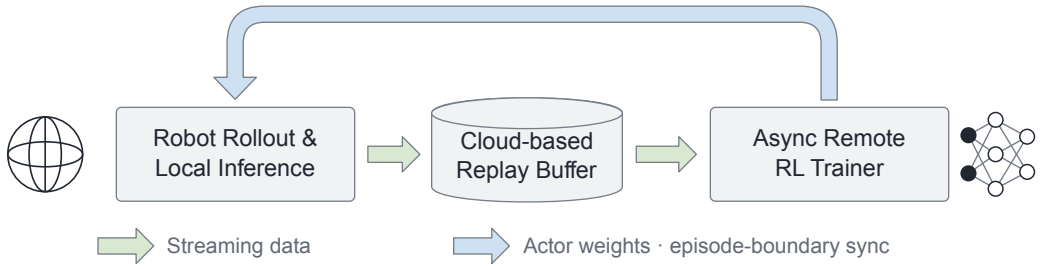


Figure 14: **Asynchronous real-world learning loop.** Green arrows denote replay-data flow from robot rollouts to the remote learner. The blue return loop denotes actor-weight delivery; new weights are activated only between episodes.

Continuous execution and aligned replay. We use real-time chunking (RTC) (Black et al., 2025c) to generate 16-step proposals while the controller continues executing buffered actions at 30 Hz. Inference takes approximately 4 control steps; these committed actions form a fixed prefix, with no soft blending beyond it. The tactile actor refines the original proposal window using fresh contact observations and cached visual-action context, without replacing already committed commands. Replay stores the resulting executed controls after chunking, refinement, and deployment projection. Training samples sliding windows $\mathbf{A}_{u:u+16}$ at arbitrary valid frames, expressed relative to each window's starting state. Human and robot trajectories therefore share a control-time grid and action convention. Overlapping windows provide more training start points, not independent additional observations.

Durable episodes and annotation alignment. Each online episode is annotated with milestone labels after it finishes. Episode arrays are stored as objects, while a relational database holds episode metadata, annotations, learning-round state, and checkpoint references. The implementation uses Cloudflare R2 for object storage and PostgreSQL through Supabase for metadata. An episode identifier is a SHA-256 digest of its arrays, shared by the uploader and annotator. Re-uploading the same episode is therefore idempotent, and annotations can be joined to an episode even if they arrive before the upload or its directory is renamed.

The uploader writes the episode objects before inserting the metadata row that makes the episode visible to the learner. An interrupted upload can leave unreferenced objects, but does not expose a partial episode through the ingestion interface. Newly available labels are reconsidered on subsequent ingestion polls. Eligible episodes are materialized into the same local layout used by the offline data-preparation pipeline, allowing the online learner to reuse that pipeline. A bounded replay window retains the newest episodes by recording time; materialized directories outside this window are removed so they cannot be rediscovered by cache builders. Recording order is preserved explicitly because it determines cache indices and therefore seeded replay sampling.

Recoverable learning rounds. A round identifies a set of newly ingested episodes and the preceding actor checkpoint. A learner claims the round under a renewable lease before preparing data and training. If that learner fails, an expired lease allows another attempt to recover the unfinished round. The lease must cover the longest uninterrupted pipeline stage and is renewed between stages.

After checkpoint creation, publishing its reference and marking the associated round complete occur in a single database transaction. This avoids declaring an episode incorporated without making the corresponding actor available. An attempt that no longer owns the round cannot successfully complete publication. The guarantee concerns consistent ingestion and publication bookkeeping: a

failed round may be recomputed, and replay intentionally revisits its samples. It is not a claim that every episode contributes to only one gradient update or that interrupted optimization is executed exactly once.

Versioned policy handoff and bounded waiting. A robot-side synchronization worker reports the actor version actually running, polls for newly published checkpoints, and downloads them in the background. A completed download atomically replaces the local checkpoint file, and the policy server loads the new actor at the next episode boundary. This separates checkpoint transfer from activation and prevents a partially downloaded file from being loaded. Fresh runs reject leftover checkpoint state; resume checks require both a checkpoint and its matching version record. Actor delivery is more frequent than critic checkpoint transfer, since critic weights are not needed for robot control.

The pre-episode gate tracks annotation availability, completion of the round incorporating that annotation, publication of its actor, and local checkpoint availability. This wait has a configured timeout, so learner unavailability need not indefinitely block the next episode; when it expires, collection may proceed using the available actor. Heartbeats report the version actually in use rather than merely the newest downloaded version. On the learner side, stale robot heartbeats or excessive lag between published and running actor versions pause further ingestion until the robot catches up. The lag check is enabled only after an initial publication, avoiding a startup dependency cycle. These mechanisms regulate progress but do not guarantee that every rollout always uses the latest weights.

Persistent feature caches and parallel updates. Training-time caching is distinct from the within-rollout key–value cache used by the tactile actor. GPU producer processes precompute frozen visual-language prefix representations for recorded frames and write per-episode shards. They reuse the trainer's prompt construction and image preprocessing, including image ordering, cropping, and padding. Completed shards can be reused on restart; incomplete shard writes are not exposed to readers. Cache validity requires the same frozen checkpoint and preprocessing, so features from before task-specific supervised adaptation cannot be reused unchanged afterward.

Sequential decoding reads each camera stream once per episode, avoiding repeated random-access video decoding. Work is distributed over frame windows so that even a single newly arrived episode can use multiple workers. Actor and critic optimization is data-parallel, with gradients explicitly averaged across ranks; equal per-rank batch sizes of B over N ranks yield an effective batch size of NB. Long data-preparation stages run separately from heartbeat and polling work, and distributed synchronization timeouts accommodate rank-specific evaluation and checkpoint writing. These choices improve utilization without changing the RL losses.

Update budget and replay composition. An online update governor tracks cumulative ingested frames and optimization steps already taken. It schedules additional steps to reach the configured update-to-data target. The frame counter is cumulative over the run, not recomputed from the currently retained replay window: otherwise the budget would stop increasing once that window filled. Episode deduplication prevents retries from counting the same arrival as new data. Scheduled work follows Algorithm 1 and the replay mixture in Equation equation 16. The update-to-data target controls optimization relative to physical experience, while the critic-to-actor update ratio balances the two optimization objectives.

Achieved update rates and rollout cadence. Let F_{new} count newly collected online control frames and let U_Q and U_π count critic and actor optimizer steps during online learning, respectively. We achieve

$$\text{UTD}_Q = \frac{U_Q}{F_{\text{new}}} = 0.4, \qquad \text{UTD}_\pi = \frac{U_\pi}{F_{\text{new}}} = 0.1. \tag{18}$$

These rates correspond to 4 critic updates and 1 actor update per 10 newly collected online control frames. The ratios count optimizer steps, not the number of replay samples processed; offline samples in the mixed minibatches do not enter the denominator. Normalizing instead by $F_{\text{new}}/16$ gives 6.4 critic and 1.6 actor updates per 16-frame chunk-equivalent. This is a change of units, not a count per actual inference call: sliding replay windows and overlapping chunk execution do not define disjoint 16-frame decision intervals.

Both RL experts contain approximately 200M parameters. Despite this trainable model size, asynchronous processing and cached prefix computation typically allow learning to complete within the

rollout-and-reset cycle. The no-additional-wait observation includes human-assisted environment reset time; it does not imply zero reset time or guarantee stall-free operation without that interval.

Consistency validation. Controlled tests compare the distributed ingestion–training–publication loop with the same training recipe run directly from the same episode sequence and initialization. Within a fixed numerical environment, matching random seeds, recording order, and sampled data permit comparison of checkpoint hashes rather than learning curves alone. Cache checks similarly compare persisted features with direct computation using identical inputs and preprocessing. Fault-injection tests exercise interrupted learners, interrupted uploads, expired leases, competing publication attempts, delayed labels, and stale heartbeats. These checks target data association, recovery, and training-loop fidelity; they do not establish manipulation performance, cross-hardware determinism, or multi-GPU scaling efficiency.